VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

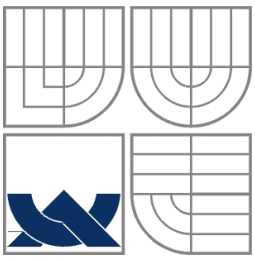
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

INTELLIGENT SYSTEM FOR GENERATING AND ANALYSIS OF TRADING RECOMMENDATIONS ON FINANCIAL MARKETS

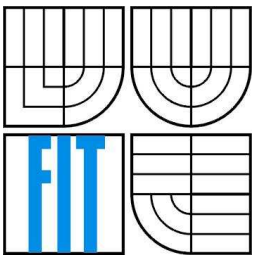
DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE ONDREJ MARTINSKÝ
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

INTELLIGENT SYSTEM FOR GENERATING AND ANALYSIS OF TRADING RECOMMENDATIONS ON FINANCIAL MARKETS

INTELIGENTNÍ SYSTÉM PRO GENEROVÁNÍ A ANALÝZU OBCHODNÍCH DOPORUČENÍ NA FINANČNÍCH TRZÍCH

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE — ONDREJ MARTINSKÝ
AUTHOR

VEDOUCÍ PRÁCE — DOC. ING. FRANTIŠEK ZBOŘIL, CSC.
SUPERVISOR

BRNO 2009

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2008/2009

Zadání diplomové práce

Řešitel: **Martinský Ondrej, Bc.**
Obor: Inteligentní systémy
Téma: **Inteligentní systém pro generování a analýzu obchodních doporučení na finančních trzích**
Kategorie: Umělá inteligence

Pokyny:

1. Seznamte se s teorií chaosu a s problematikou fraktální geometrie. Prostudujte teorii efektivních trhů, principy technické analýzy vývoje cen akcií a problematiku money-managementu při obchodování na burze.
2. Prostudujte problematiku soft-computingu, zejména problematiku neuronových sítí, fuzzy systémů a evolučních algoritmů.
3. Navrhněte systém, který bude inteligentně zpracovávat data z burzy. Systém by měl pracovat adaptivně, na základě daných podmínek by měl vyhledávat vhodné investiční příležitosti a na základě daného money-managementu by měl generovat vhodná investiční doporučení.
4. Navržený systém implementujte ve zvoleném prostředí (např. C nebo Java).
5. Systém otestujte na historických, případně i aktuálních datech z burzy. Simulací určete jeho výnosnost.

Literatura:

- Plummer T.: Forecasting Financial Markets, Kogan Page Ltd, 2008, ISBN 9-78-0749452261
- Kaufman P.: New Trading Systems and Methods, Wiley Ltd, 2005, ISBN 978-0471268475
- Kirkpatrick C., Dahlquist J.: Technical Analysis: The Complete Resource for Financial Market Technicians, FT Press, 2006, ISBN 0-13-153113-1
- Russel S., Norvig P.: Artificial Intelligence, a Modern Approach, Pearson Education Inc., 2003, ISBN 0-13-080302-2
- Fishwick P.: Simulation Model Design and Execution, Prentice Hall, 1995, ISBN 0-13-098609-7
- Zeigler B. P.: Theory of Modeling and Simulation, Academic Press; 2 edition (March 15, 2000), ISBN 978-0127784557

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Zbořil František V., doc. Ing., CSc.**, UITS FIT VUT
Datum zadání: 22. září 2008
Datum odevzdání: 26. května 2009

L.S. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ Fakulta informačních technologií Ústav inteligentních systémů 612 66 Brno, Božetěchova 2

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Práca sa zaoberá problematikou vývoja cien finančných inštrumentov v kontexte správania sa davu obvplyvneného davovou psychózou. Práca dáva do kontrastu fenomén davovej psychózy a teóriu efektívnych trhov, ktorá vo svojej podstate vychádza zo všeobecnej ekonomickej teórie. Základným predpokladom je fakt, že pokiaľ existuje model vzájomnej interakcie medzi jednotlivcami na trhu, potom existuje aj prostriedok pre predikciu správania sa trhu ako celku nezávisle od toho, že správanie jednotlivcov samo o sebe predikovateľné nie je. Z praktického pohľadu práca popisuje prostriedky technickej analýzy vývoja cien finančných aktív a pojednáva o možnostiach využitia soft computingu pri konštrukcii automatických obchodných systémov, ktoré dokážu analyzovať situáciu na finančnom trhu a generovať krátkodobé obchodné odporúčania. Práca kombinuje klasické obchodné stratégie založené na indikátoroch s fuzzy logikou, a skúma možnosti použitia genetických algoritmov pri optimalizácii obchodných stratégií. Súčasťou praktickej časti práce je aj framework pre zostavovanie, simuláciu a hodnotenie obchodných stratégií. Vďaka tomu že simulátor je založený na formalizme DEVS (Discrete Event System Specification), v prípade potreby nie je problém zaviesť do modelu komponenty pracujúce v reálnom čase. Súčasťou práce je taktiež databáza historických dát a nástroje pre jej automatickú aktualizáciu.

Klíčová slova: Behaviorálna ekonómia, Technická analýza, Fuzzy logika, Genetické algoritmy.

Abstract

This work deals with problematic of market price prediction in the context of crowd behavior affected by the mass psychology. The work highlights the contrast between a phenomenon of mass psychology and the efficient market hypothesis, which is essentially based on a common economic theory. The basic assumption is that if there is a model of interaction between masses and agents participating in markets, then there also exist means for prediction of the whole market behavior, nevertheless that the behavior of every single agent is not predictable. From the practical point of view, this work describes methods of technical analysis used for a prediction of price movements and discusses a soft computing approach used in a composition of automated trading systems. This work combines classical trading strategies based on indicators with fuzzy logic, and deals with genetic algorithms used for optimization of such strategies. The practical part of this work also contains a framework for composing, simulation and analysis of the automated trading strategies. The simulator contained in this framework is based on DEVS formalism (Discrete Event System Specification). Because of this, there is a possibility to embed real-time components into the trading model. This work contains also a database of historical financial data and tools for their automatic actualization.

Key Words: Behavioral Economics, Technical analysis, Fuzzy logic, Genetic algorithms.

Bibliographical citation

Ondrej Martinský: Intelligent System for Generating and Analysis of Trading Recommendations on Financial Markets, master thesis, FIT VUT Brno, 2009

Declaration

Hereby I declare that I wrote this thesis myself with the help of no more than the mentioned literature and auxiliary means.

.....
Ondrej Martinský

Acknowledgement

The author is indebted to the supervisor of this thesis, doc. Ing. František Zbořil, CSc. for his great help.

Contents

List of Figures

Introduction

Over the past years, we have an opportunity to see the difference in evolution of trends in information technologies. Today, information technologies are not an independent science discipline, but they have penetrated into all aspects of modern life and became interdisciplinary matter. Because of this, we also see some differences between the reality itself and information technologies, which are primarily aimed to reflect this reality in a proper and utilizable way. Information technologies as a part of exact science are in a strong contrast with a real environment, which is inexact, uncertain and indeterminable. Applications based on an exact and symbolic representation of the reality bring some difficulties to their interaction with the surrounding real environment. Because of this, the soft computing methods have gained on importance in last years.

Optimization of artificial intelligence-based applications for a real usage is nowadays one of the biggest technical challenges. Such optimization is a key measure for successful treatment with the uncertainty of the reality. Laboratory conditions in which applications are developed often don't correspond to an environment in which the applications will be finally settled. Unlike the traditional hard computing systems and besides the appropriate application structure, there is also a need to find appropriate parameters of a system and, of course appropriate evaluation metrics. The discussion of how well an intelligent system should meet customer's requirements is also a source of ambiguity. The functionality of classical hard computing applications is evaluated only in the context of strictly defined demands and constraints, but even theoretically well-made intelligent systems often don't match real customers' needs and expectations.

Generally, the real world is a fight in many aspects, and one of them is a fight for money in a free market. In the world financial marketplace, agents are trading with each other and challenging with a system, whose part they actually are. The purpose of this work is to describe the behavior of masses with the soft computing methods and to build a profitable intelligent system which would practically gain from the theoretical knowledge of the market behavior.

The neoclassical economic theory and the efficient market hypothesis assumes that the behavior of the whole market is given as a sum of rationally reasoning agents as well as the price in a free market exactly reflects its fundamental situation. In fact, the reasoning of agents acting on financial markets is not rational, but agents influent each other in terms of the mass psychology. Because of this, the behavior of masses has its own psychological attributes and thus it is predictable – even by the artificial intelligence.

The good question is if a system based on the artificial intelligence is able to analyze market behavior more precisely than a flesh-and-blood trader, and to generate more reliable trading recommendations. If the trader evaluates market behavior detachedly, he is able to take much better decisions and achieve much better trading results. The point is that traders are rarely psychologically detached from decisions which they take, because they are handling with their own money, and taken decisions markedly affect their wealth. In other words, the mental blocks often disallow traders to take rational decisions. Because of this, the trading is considered as intellectually simple, but psychologically difficult activity. The system based on the artificial intelligence would be the right choice for this kind of activity, because it lacks psychological aspects of the human nature.

From the system theory point of view, a crowd is a self-organizing hierarchical system, which is describable and predictable by computation techniques of the artificial intelligence and soft computing. Since an intelligent system stands out of the mass, it can be a successful tool for the prediction of market behavior and for detection of advisable trading opportunities. In this field of research, this work deals with a development, implementation and simulation of appropriate trading strategies as well as with the money and risk management methods for securing speculative operations with financial derivatives.

This thesis is organized into two parts. The first one contains three chapters and defines the problem from an economic point of view. The second part, also with three chapters, discusses possible solutions from the technical and mathematical points of view. In other words, the main purpose of this thesis is to apply mathematical and artificial intelligence methods to the economic problem.

This work is intended for all who are concerned in the topics of trading, financial markets and security exchanges as well as for all who have either theoretical or practical knowledge from the fields of artificial intelligence and soft computing, wondering how these topics can be applied in financial markets.

The first chapter introduces readers to various economic theories and hypotheses, such as to the efficient market hypothesis, chaos theory and to the behavioral market theory. The main intention of the first chapter is to comparatively illustrate contradictions in these theories and to show that the economics as a whole is not an exact science. The conclusion of the first chapter is based on an assumption that although these theories are in the contradiction, the truth is somewhere at the intersection of them and the exact tenets depend on a purpose and field of the study.

The second chapter describes the dynamics of crowd behavior. The basis premise is that the whole society (or, respectively, any crowd) is a system of interconnected agents. However, entities and relationships between them are enormously complicated. There are two points of view. Individual humans can be studied by the cognitive psychology and the society as a whole by the system theory. The second chapter classifies society on a financial market as a dynamic non-linear self-organizing system and models it by limit cycles, energetic barriers and information shocks.

The third chapter introduces readers to the basic tenets of automated trading systems and to fundamentals of the money management and statistical evaluation of trading results. The visual analysis and interpretation of Elliott waves introduced in the second chapter is a very subjective process. Ambiguities which occur in charts may cause that traders firstly perform a deep technical analysis and then they make the final decision and final interpretation only on a psychological basis. This problem doesn't exist in mechanical trading systems, which generate exact (psychologically unbiased) advices for traders. The advantage of the mechanical trading systems is that they provide crisp signals and eliminate psychological aspects of trading. The third chapter introduces readers also to several trading strategies based on various types of indicators and shows how the statistical analysis can be used to evaluate income, stability and robustness of these strategies. Although the third chapter explains how trading strategies generate recommendations on markets and how to statistically evaluate the outcome of such strategies, it doesn't make readers familiar with the principles of such evaluations.

Because of this, the fourth chapter discusses several aspects of the simulation of trading environment and proposes several models of time and price skews that occur on markets. It introduces readers to the basics of the Discrete Event System Specification and constructs an appropriate trading simulator.

The fourth chapter provides also some technical knowledge and resources needed for the simulation of trading strategies proposed in the third chapter and contains a very simple simulation case study. Seeing that the trading simulator doesn't fully respect the specification of DEVS, this chapter presents all customizations and explains why they are important for more accurate simulation results.

Since simulation tools allow traders to repeatedly and comparatively run various strategies on the same data, they are appropriate also for a wide spectrum of optimization tasks. The optimization of trading strategies is a process of making them more profitable, robust and stable considering given market conditions. Because of this, the fifth chapter introduces readers to the parametric trading strategies and shows how the genetic algorithms can be used for the optimization of them. The fifth chapter contains also a case study of the genetic optimization and explains the biggest problem of all optimization methods, which is the curve fitting of in-sample market data.

Automated trading systems are based mainly on a computational evaluation of crisp inputs. Even though this evaluation is an objective process that eliminates psychological aspects of trading, it is not very robust since it doesn't reflect well the vagueness of the real trading environment. This is caused by a fact that conventional automated trading strategies are primarily based on an exact and symbolic representation of the reality.

Several decades ago, scientists and researchers believed that every problem, no matter how complex it is, can be solved by a computational system with an adequate power. The uncertainty found everywhere in nature forced scientists to reevaluate their approaches and attitudes to solving large, complex problems. The concept of uncertainty originates from the shift of paradigm in the artificial intelligence. If we are not able to avoid the uncertainty, we must use computational methods which implicitly allow an uncertainty in the input data (for example like the soft computing approach do).

Because of this, the sixth chapter shows how the soft computing, especially fuzzy logic, can be used for a more reliable prediction and modeling of the market behavior. This chapter explains general principles of fuzzy logic and fuzzy reasoning and shows how we can use fuzzy controllers to eliminate the sensitivity of signaling systems in trading strategies.

The sixth chapter also introduces readers to the Triple Screen Trading System and proposes a way how this system can be improved by a fuzzy inference mechanism. Additionally, the sixth chapter analyzes sensitivities of both conventional and fuzzy-based triple screen trading systems and compares their characteristic properties, such as the robustness, stability and the overall outcome.

Chapter 1

Reality, the intersection of multiple theories

Generally, the common economic theory is not an exact science, since it describes relations between real individuals who aren't exact too. Because of this, various theories and hypotheses have been introduced for description of the market behavior. Some of them are expecting that agents acting on financial markets are rationally reasoning, and other employ a phenomenon of the mass psychology for better understanding of irrational behavior of crowds.

The economic theories and theories of market behavior are regarded as only approximating models of the real markets. Because of this, they are still under the development process. The occurrence of events on financial markets that haven't precedents in the history forces us again and again to revaluate our former approximations, models and theories of the behavior in economics. Because of this, various contrary theories have been evolved during the past decades, but none of them represents the reality with adequate confidence. The best representation of the reality is somewhere at the intersection of multiple economic theories, but it is hard to say how well one or other theory describes it.

In addition, economic theories and models affect the reality itself, because they affect behavior of masses which make decisions based on these models. This is the answer to question why new and new events without precedents are still occurring on markets. The example of a theory which is applicable backwardly in the past, but doesn't work in current times is the Dow Theory. This fact is simply caused by the investors, who modified their behavior after this theory has been published.

This chapter introduces several theories of market behavior to illustrate the inexactness of the whole economic science. This is the key observation that helps us to understand how the technical analysis describes the behavior of crowds on financial markets and how to utilize these descriptions in the construction of profitable intelligent systems.

1.1 Efficient market hypothesis

The efficient market hypothesis is derived from the well-known theory of rational choice, which is based on formal models of the social and economic behavior. The theory of rational choice together with the neoclassical synthesis is a major theoretical paradigm in the modern economic science. This theory is based on an assumption that every single agent on the market has its own preference function indicating the utilization of combination of goods, and he is performing rational and effective actions to maximize this utilization. The theory of rational choice provide us models which don't fully describe the reality, but these models help us to take decisions which can be considered as rational in the context of maximization of the good utilization.

The efficient market hypothesis assumes that financial markets are informationally efficient. This means that all known information is actually involved in the current market price,

thus it is impossible to continually overperform average market revenues by trading with public information. According to some thoughts, any relevant information that can affect the market is flashed to it within thirty seconds.[1]

There is one common misleading fact about the efficient market hypothesis. Compared to the theory of rational choice, the efficient market hypothesis doesn't require agents acting on the market to be rationally reasoning. When new information is published, some traders may underrate it, and others can overrate it. However, individual reactions of traders are counted together, and the net market price change is then given as a sum of the behavior of a big number of irrationally reasoning agents. Since particular mistakes of overrating and underrating traders are mutually eliminated, the behavior of the whole market is always right and corresponds to published fundaments.

Three forms of this hypothesis have been defined according to its strength. (1) The *weak form* declares that extraordinary returns cannot be earned by using strategies that rely on historical movements of the prices. Because of this, the technical analysis and methods based on a serial dependence between market prices cannot be used for a prediction of the future prices. This form is referred as weak, because it supposes that publicly available fundamental information is not immediately flashed into the current prices, thus it can be used for a prediction of the short-term market movements.

The *semi-strong form* of this hypothesis declares that continuous extraordinary returns cannot be gained using publicly available information, because the price is adjusted according to published information immediately. Since there is no enough time to realize market transactions between when the information is published and the market is adapted, there is also no way to utilize this information. However, the semi-strong hypothesis allows insider traders to use non-public information for achieving profits before markets can adapt to it.

A *strong form* of the efficient market hypothesis assumes that all public and private information is actually involved in the market prices, thus it is impossible even for insider traders to continually overperform average market returns. The strong form explains the existence of a small group of investors that excessively and continually overperform market returns as a result of the normal distribution of earnings over the huge number of individuals.[2]

According to our previous experiences, we agree that fundamental events affect market prices immediately. However, we think that the market as a whole often overrates published information, and because of this, the prices are often irrational. The efficient market hypothesis doesn't provide explanation of market bubbles, crashes and speculative fluctuations of market

[1] The information about the disaster of Space Shuttle Columbia was flashed to market prices within nine seconds after it had been published.

[2] Warren Buffett explained this fact in his talk given at Columbia University in 1984 using the model of "coin-flipping contest". We assume that every day 225 million Americans wager a dollar and if they call correctly, they win a dollar back from those who called wrong. After twenty days there will be 215 people, who called their coins flip twenty times correctly in a row. Each of these people turned one dollar into a million. The reasoning of this example supports the explanation given in the efficient market hypothesis, but Warren Buffett argued that there are some differences in a distribution between winners in this example and the successful investing professionals. There is nothing surprising on the example of coin-flipping context, but it would be suspicious if all winners came from the same village. This is exactly what is happening on financial markets – a small group of extraordinary successful investors share several common characteristics and strategies. These strategies are based on long-term investments, deep fundamental analysis of securities and on a concept of the market pricing instead of market timing.

prices. These fluctuations are caused by similar behavior of a big amount of individuals at the same time. If this kind of mass behavior is rational, then there must be rational well known fundamental reasons which support it. The history of bubbles and crashes reveals that no such reasons were available. Instead of this, investors often assure each other that a certain type of investment is a "sure thing", because it is continually rising. The 1637's Tulip mania[3], 1720's South Sea Bubble[4] or the 2000's dot-com bubble are just practical disproofs of the efficient market hypothesis.

1.2 The theory of chaos

Scientists define the financial market as a complex self-organizing hierarchical non-linear dynamic system. The keywords *non-linear* and *dynamic* are important in the context of the theory of chaos. A non-linear system is the system which involves more complex than a linear dependence between input and output variables. The keyword *dynamic* means that the system is evolving in the time. The chaos theory studies behavior of natural systems from many fields of the science and examines characteristics which these systems share. One of the common characteristics of these systems is a sensitivity to input conditions. A very little change in the input conditions can cause a big deviation of the system's behavior over a period of time. This phenomenon is also known as a butterfly effect.[5] Even the systems are deterministic in their structures; they seem to be non-deterministic due to a complex behavior caused by the butterfly effect. The whole universe is considered as non-deterministic just for this reason.

One of the fields where the chaos theory is applied is a research of financial markets and economics. According to this theory, it is impossible to predict future market prices by applying patterns occurred in the past to a current situation on the market. This is caused by several reasons. One of these reasons is that the current market situation is defined by thousands of various parameters, such as the following:

- Horizontal and vertical structure of the crowd.
- Traders' beliefs, hopes and desires.
- Market prices, traded volumes and open interest.
- Fundamental situation in economics.

It is impossible to comprehensively describe the whole market situation due to not all parameters are quantitative as well as there is no way how to observe them. Usually, we know only the market price and the volume of realized transactions. Because of limited amount of known parameters, we wouldn't be able to predict behavior of a market system even if it was deterministic and non-chaotic.

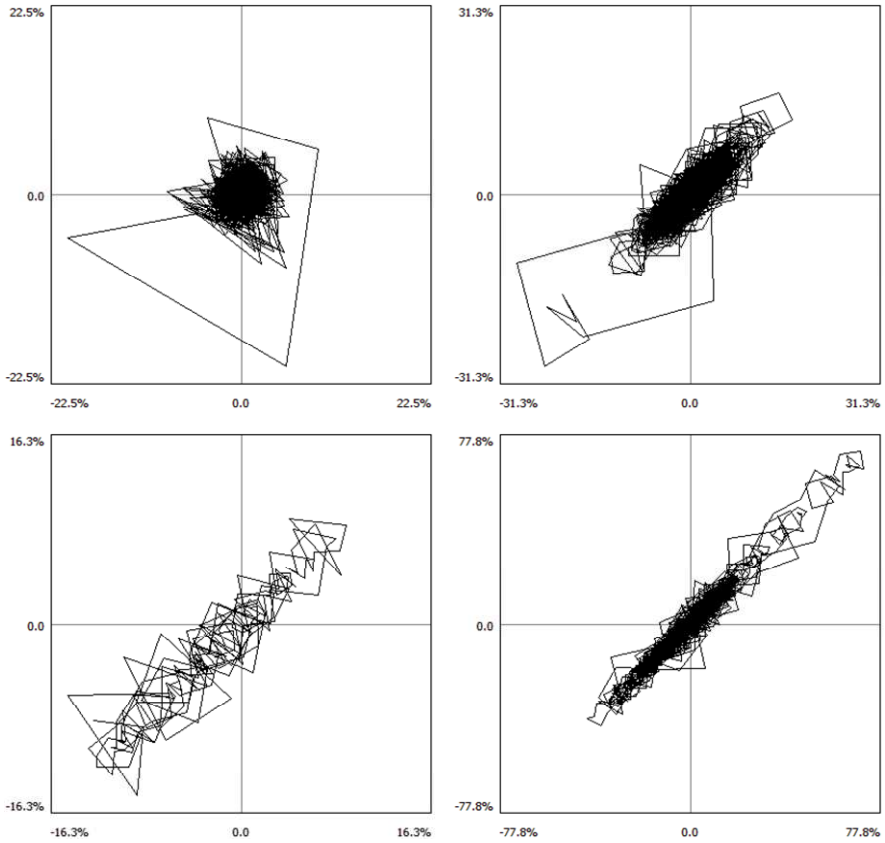
[3] Tulip mania was a period during which the prices of future contracts for tulips rose to extremely high levels and then (un)expectedly collapsed.

[4] The South Sea bubble was a bubble in stock prices of The South Sea Company. The underlying cause of this bubble was a monopoly right to trade with South Africa.

[5] Even a small change in the atmosphere caused by a butterfly's wing can evoke a tornado on the opposite side of the world.

Some systems are chaotic in all phases, but most of them are chaotic only in a certain subset of the phase space. The state of a system at the specific time is represented by a point in the n-dimensional parameter space. The set of points to which the system converges after a long period of time is referred as *attractor*.

Let's define a two-dimensional phase space c_t / c_{t-1}, which contains pairs of daily consecutive market price changes. The daily price change is defined as a percentage difference between closing prices of two consecutive days. (2) Figure 1.1.a represents the attractor of consecutive price changes of the S&P 500 stock index between years 1950 and 2008 on a daily basis. According to this attractor, there is no significant serial dependence between the daily market price changes. In other words, short-term movements of stock prices are chaotic. Furthermore, the bounds of this attractor represent maximum daily deviations between the closing prices. As you can see in the Figure 1.1.a, the S&P 500 index hasn't exceeded a 10% daily price change over the long period of time, except the big fallout on Black Monday, October 19, 1987 when markets unexpectedly lowered over the 20%. The attractor also shows that the stock market has quickly recovered from this fallout and returned to the ordinary state within in several days.



A | B
C | D

Figure 1.1: (a) The attractor of the S&P 500 index's daily price changes between years 1950 and 2008. **(b,c)** Weekly and monthly price changes of the S&P 500 index. **(d)** Monthly price changes of the Dow Jones Industrial Average Index between years 1928 and 2008.

We have illustrated that short-term movements of the S&P 500 index involve chaotic behavior. Let's examine if the chaotic behavior is employed also in the middle-term and long-term

movements. We will consider pairs of the market price changes based on a longer time period. Figures 1.1.b and 1.1.c represent attractors of the week and month market price changes. As we can see, these attractors are slightly stretched along the diagonal line. This anomaly represents one important implication between the consecutive market prices. If a market is currently rising and the price change between today and last week is positive, then there is a big probability that the tomorrow's price change will be also positive (and vice versa for falling markets).[6]

The degree of chaotic behavior in market prices can be measured by a so-called *Fractal Efficiency Ratio*. This ratio is given as a net price change over n bars divided by the sum of individual bars' ranges (see Eq. 1.1). In other words, the volatility over n periods is divided by the aggregated volatilities of individual bars. If this ratio is equal to one, the corresponding system is non-chaotic and has a great trending characteristic. If it is equal to zero, the system doesn't have trending capabilities, thus it is purely chaotic.

$$\mathrm{FER}_n = \frac{\left|c_t - c_{t-n}\right|}{\sum_{i=0}^{n}\left|c_{t-i} - c_{t-i-1}\right|} \tag{1.1}$$

We can form several conclusions according to the attractors based on various time frames of the same market. The shape of these attractors tells us how well the market is able to trend, and the bounds of represent the strength of market price movements. The bigger $\mathrm{FER_n}$ ratios correspond to more stretched attractors. By comparing attractors based on daily and weekly time frames, we assume that daily price changes are chaotic and unpredictable, but weekly price changes beside the chaotic behavior include also trending tendencies. This fact is in the contradiction with the butterfly effect, because we are able to predict price movements on a weekly basis even if the movements on a daily basis are purely chaotic.[7]

If the market price movement is not purely chaotic, there must be a pattern or a model which describes it. The market researchers discovered that these patterns exist in a chart based on an arbitrary time frame. For example, patterns based on a daily time frame are formed together into weekly patterns, then the weekly patterns are formed into the monthly ones, and so on. This concept resembles the well known fractal geometry, which is also based on an assumption that several low-level geometrical patterns form the higher one.

Consider the measurement of a state border line. The length measured on a big scale map will be different than the length measured more precisely on a detailed plan. The more precise measurement involves more detailed shapes in the line. Because of this, the length of such line depends on a measurement and grows with the increasing degree of details. In extreme case, when we omit the atomic integrity, the length will go to infinity.

Since price movements share several properties with the fractal geometry, the estimation of a next price change also depends on a scale of the measurement. Because of this, the estimate

[6] Points in the upper right quadrant of the attractor mean that both today's and yesterday's price changes are positive, thus market is continually rising. Points in the bottom left quadrant represent falling market. There are also several points in the upper left and bottom right quadrants. These points represent situations where market changes its direction.

[7] The weekly time frame of an attractor means that the percentage change of closing prices is computed on a period of one week, independently that the time frame of the underlying chart is one day.

of a next price movement will be always different from the reality, no matter how deep we will go in the level of details during the estimation.

1.3 Behavioral market theory

The behavioral market theory is a theoretical background for a wide range of technical analysis methods. This theory is a part of the behavioral economics, which applies scientific research on human behavior affected by emotional factors.[8]

There are two different points of view. The first view deals with emotionally affected individuals. The reasoning of individuals is studied by the cognitive psychology. The second view is covered by the behavioral market theory, which studies behavior of the crowd composed of such individuals.

We discussed several aspects of the efficient market hypothesis in Section 1.1. The tenets of the behavioral market theory are in the strong contrast with the principles of efficient markets. Although both theories tolerate an existence of the irrationally reasoning agents, the efficient market hypothesis assumes that irrational reactions of traders are mutually eliminated. The important difference between these theories is that the behavioral market theory doesn't require such assumption.

The efficient market hypothesis and the neoclassical economics define rationally reasoning individual (Homo Economicus) as a calculating unemotional utilization maximizer. The defenders of the efficient market hypothesis argue that the behavior of financial crowd is altered by the principles of competition and evolution, thus individuals who were unable to act rationally had been eliminated by the evolutionary forces. The biggest shortcoming of the neoclassical economics is that it completely ignores cognitive and psychological aspects of economic agents. This is particularly caused by a fact that rational economic framework is easier to formalize. The behavioral economics tries to formalize even the psychological aspects, which is much more difficult objective.

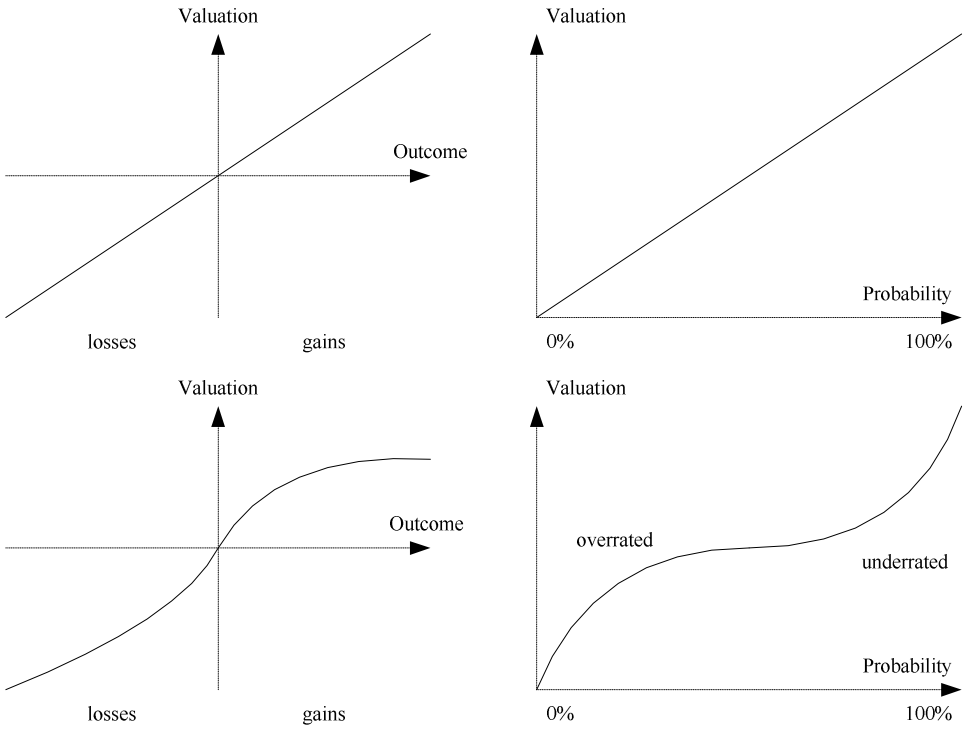
Professors S. Mullainathan and R. H. Thaler in their work *Behavioral Economics* establish three different ways in which human deviates from the standard model of Homo Economicus. The first is a *bounded rationality* which reflects limited cognitive abilities of humans to solve problems. The second deviation is a *bounded willpower*, which is based on a fact that people often make choices that do not correspond to their long-term interest. For example, people often prematurely take their money out of mutual funds in bad times when the security prices are (temporarily) low. The *bounded self-interest*, as a third deviation, stands on an assumption that humans are often willing to defer their own interests to help others.

Even if people have sufficient time and resources needed for their reasoning, they will never be rational until they are emotionally related to decisions which they take. Because of psychological self-control reasons, humans often refuse to choose the rational solution, even if they know it. For most people, the short-term psychological satisfaction is more important than the long-term financial gain.

One of the pylons of the behavioral economics is a *prospect theory* (3), which describes how humans make decisions between alternatives which involve risk. Compared to the expected

[8] S. Mullainathan and R. H. Thaler in (1) defined behavioral economics as the combination of economics and phychology, which investigates what happens on the market in which some of the individuals display human limitations and complications.

utility hypothesis[9], the prospect theory regards also psychological aspects of human decisions. Let's assume that humans quantitatively evaluate the objective outcome and the probability factor in each decision. Rationally reasoning agent may evaluate the outcome and the probability "as is". Because of this, there should be a continual proportion between the outcome and the probability itself and the fact how humans subjectively evaluate it (Fig. 1.2.a and 1.2.b).



A B
C D

Figure 1.2: (a,b) Rationally reasoning agent valuates outcomes and probabilities objectively. **(c,d)** Psychological aspects of the human nature deform this valuation. The result is a loss aversion and the tendency to overrate and underrate events.

Since the psychology of human being deforms objective econometrical dependencies, the prospect theory has been presented for a better understanding the real human behavior. There are several causes why people don't evaluate uncertain outcomes objectively. Since losses have a bigger psychological impact than gains, many people have a tendency to prefer avoiding losses than acquiring gains. Because of this, the curve representing how people evaluate the objective outcome is not symmetrical (Fig. 1.2.c). The theory of bounded rationality shows that people are more sensitive to changes of wealth rather to an absolute level of it. This is represented by a concave shape of the outcome function. The probability and the risk are also not valuated objectively. In general, people have a tendency to underrate events with the big probability and to overrate the small ones.

Let's assume that we have n opportunities, each with the corresponding outcome and probability. The utilization of such outcomes is given by the following equation:

[9] The expected utility hypothesis mathematically describes betting preferences of people with regard to risky and uncertain outcomes. It is strictly rational and it does not consider psychological aspects of human decisions. Because of this, the expected utility hypothesis is widely accepted as a model of the rational choice theory.

$$U = \sum_{i=0}^{n} v_o(o_i) \cdot v_p(p_i), \tag{1.2}$$

where o_i is the objective outcome of the i^{th} opportunity, and p_i is the objective probability of such outcome. Since humans don't valuate outcomes and probabilities objectively, this formula employs also two functions for subjective psychological valuation. Function $v_o(o)$ is the subjective valuation of the outcome o, and function $v_p(p)$ is the subjective valuation of the outcome probability p.

The prospect theory is an econometrical illustration of ambiguities between the objective valuation of uncertain outcomes and the way how humans subjectively valuate them. Besides the prospect theory, there are many other evidences which proof that ordinary econometrical methods based on a rational reasoning don't work in real markets. The purpose of the psychologically adjusted models is not to provide a formalized framework for estimations of market prices, but only to help understand how real humans behave and why technical analysis methods work.

The purpose of this chapter was to introduce several economic theories and to highlight several major differences and contradictions between them. Since the economics is not an exact science, the contradictions only support the fact that real market behavior cannot be fully described by theoretical background. As the title of this chapter suggests, the reality is somewhere at the intersection of multiple theories.

Chapter 2

The dynamics of crowd behavior

The previous chapter discusses various economic theories, which are necessary to understand basic principles of the crowd behavior. This chapter analyzes crowds from the system theory point of view and explains their structure and behavior in a context of the energy and information exchange. This chapter also introduces readers the theory of co-evolution, limit cycles and the hierarchical organization of crowds and explains why markets are always oscillating up and down, nevertheless no relevant information related to them is published. All of this helps us to better understand the theory of Elliott waves.[10]

2.1 The system theory point of view

The system theory helps us to understand why oscillations are periodically occurring on markets without any prior fundamental causes or reasons. The system theory is a relatively complex approach that tries to describe and to understand nature laws by studying processes rather than structures or static settlements. Because of this, the system theory treats every organism in the nature as a self-organizing hierarchical system. Each such system tries to keep its balance with the surrounding environment by the information and energy exchange. Since this, we are able to study crowds on financial markets either in the context of natural or social sciences.

The term *self organizing* means that the system has an ability to reestablish its balance in a case when it is disrupted by an external event. A crowd (as such system) is formed in a reaction to the changes in the environment. Individuals often join the crowd to defend themselves against the common threats. In financial markets, such threat can be the loss of money or the waste of profit opportunities. The crowd affects the surrounding environment until it achieves established goals. In some cases, crowds are forced to defer previous goals and to establish other ones. Every crowd exists only if all its members have the same goals. If there are no such goals, there is no crowd. This concept will be explained in more detail later.

2.1.1 The exchange of energy and information

There are two necessary system conditions which a crowd must meet to achieve its goals. The crowd must be opened for the exchange of energy and information. The energy of a crowd is

[10] The theory of Elliott waves is a complex and most comprehensive technical analysis tool for a description of market behavior published by Ralph Nelson Elliott in 1938. (22) Although not all observations described in the Elliott's work are theoretically explained, some of them are particularly covered by the underlying theories. (2)

represented by individuals who form the crowd. When an individual becomes a member of the crowd, then the energy of such crowd rises and vice versa. As every natural system, the crowd has also a tendency to minimize its entropy. Because of this, a crowd often disintegrates if there are no new events and threats that would stimulate its integration tendencies.[11]

The second necessary condition for the existence of a crowd is the exchange of information. The only way how a crowd can achieve its goals is the manipulation of the environment using the information exchange. Because of this, a bidirectional transmission of information between the crowd and the surrounding environment is necessary. Every individual in a crowd is able to transmit information as well as every individual has an ability to manipulate public opinion within the crowd. Individuals in the crowd are not equal and not every individual has the equal impact on the public opinion. Information received from the environment is in many crowds interpreted by the crowd leaders. Because of this, the principle of such transmission is a very complex and hard-describable problem, especially if the crowd leader has a very big impact on the public opinion.

Generally, members of a crowd can be easily manipulated and the information can be propagated not only between the crowd and the environment, but also between individuals within the crowd. In addition, propagated information is often altered due to misunderstanding and different interpretation. Information can be altered in two different ways during the propagation within a crowd. Some of the individuals have a tendency to overreact it, whereas others incline to underreactions. The system theory treats the crowd as a system of interconnected entities. Entities represent members of the crowd and interconnections represent information channels between them. Because of this, overreacting and underreacting tendencies of individuals are represented by positive and negative relations between corresponding entities.

Consider the example illustrated in Figure 2.1.a. (2) The information loop between individuals represented by entities A, B and C is positive, thus these individuals influence each other in a positive way and mutually confirm their assumptions and beliefs. This leads to a state, when the crowd as a whole is unreasonably sure about its beliefs. Figure 2.1.b illustrates the negative information loop between entities. Individuals represented by entities B and C attenuate information propagated within the crowd. Because of this, the crowd as a whole has a tendency to neutralize its assumptions.

The combination of positive and negative loops in a system leads to a complex oscillating behavior of a crowd. Although negative loops neutralize the evolution of crowd, it doesn't necessarily mean that the behavior of a crowd will be stable in all cases. The stability of the crowd is determined by a proportion between its progressive and conservative parts. Every conflict between the reality and the crowd's beliefs evokes compensating reactions. Crowds often eliminate these disproportions either by actuating to the environment or by modifying their own structures and goals.

Although these principles seem to be complicated, they can be easily illustrated on the following example. A crowd in financial market is represented by a group of speculators who hold similar securities and have similar planning horizon. For example, all speculators holding derivatives of the same security in a long position, in the same time and for the same time horizon are considered to be members of the same crowd. The common threat for this crowd is a loss of money caused by the depreciation of prices and the common motivation is a gain resulting from the growing prices. As new traders enter positions by buying securities, the

[11] Tony Plummer in his work (2) illustrates this concept on spontaneous street riots, which rarely persist for a longer period of time, even with the absence of police enforcement.

energy of such crowd rises. The belief of the crowd is based on an assumption that underlying security prices will be growing up in the near future.

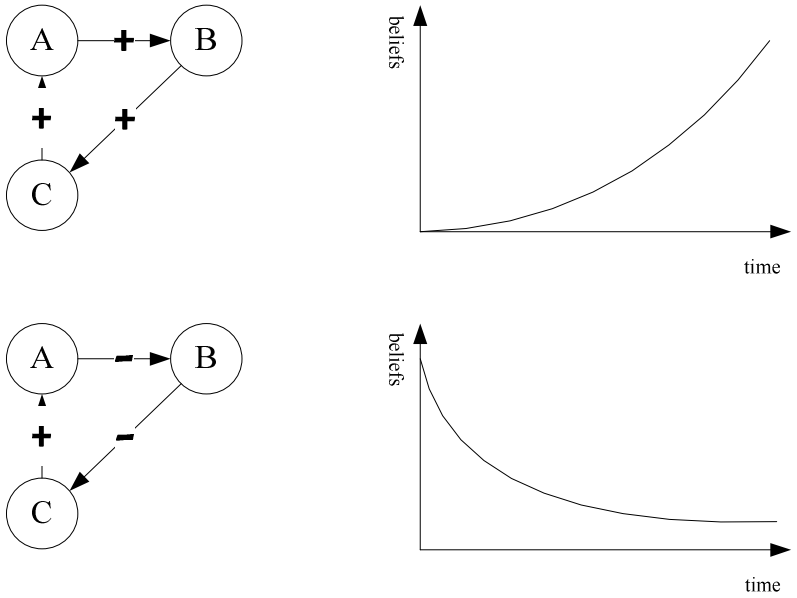


Figure 2.1: The positive **(a)** and negative **(b)** information loops between individuals in a crowd.

When members of the crowd buy new securities, they push security prices up. Similarly, when the prices are rising, new and new individuals are stimulated to enter the crowd by opening new positions. Because of this, the crowd actuates to an environment as well as the environment actuates to the crowd. This principle is often referred as a theory of the co-evolution and it has been firstly mentioned by A. Lotka and V. Volterra. Even though the original theory incorporates interactions between predators and preys, we can substitute predators by a crowd and preys by an environment.[12] In addition, hierarchical systems such as food webs usually involve individuals on several existence levels. This means that higher levels act as an environment for lower levels. After such generalization, this concept can be successfully applied also to financial markets.

2.1.2 The crowd's life cycle

We have mentioned that every system has a tendency to eliminate differences between it and the surrounding environment (or a parent system). Also, the negative information loops cause that systems involve oscillating behavior. Because of this, every crowd tries to oscillate in a harmony with the parent system. Such oscillation is referred as a *limit cycle*. If the environment

[12] Lotka-Volterra's model of predators and preys is a relatively simple model analytically expressed by differential equations and empirically verified using the data of the population dynamics of lynxes and hares collected by the Hudson Bay Company. Even though financial markets share similar principles, they are incomparably more complicated than this simple model.

or the parent system oscillates regularly, the limit cycle of a crowd will be also regular.[13] We will use a single-variable abstraction of the crowd's state to illustrate its relationship with the environment. Let x to be the state of a crowd (optimistic or pessimistic expectations), and y to be the state of an environment (percentage change in prices). The limit cycle is defined as a limited periodical oscillation between two variables and it is graphically illustrated by a limited non-linear path. (4) Figure 2.2 contains the limit cycle which illustrates relationship between the market price changes and the traders' expectations for the near future.

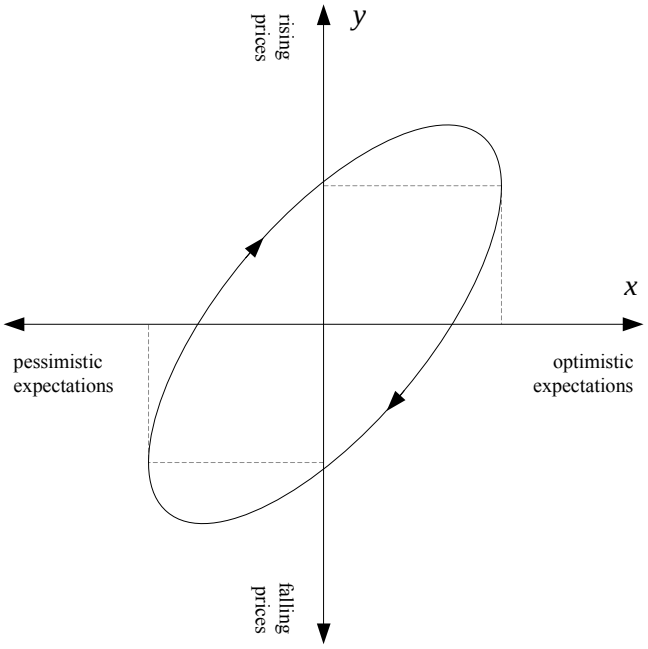


Figure 2.2: The limit cycle and the relationship between market price changes and traders' expectations.

The asymmetric shape of the cycle and dashed lines indicate one extremely important thing common for all financial markets. When traders' expectations of the future price movements are very optimistic and everyone believes that "prices will grow up to the sky", the market is usually at the end of the trend. It means that although market prices are still growing, a velocity of the growth is lower than a velocity in the middle of the trend. This principle is valid also for falling markets and negative traders' expectations. The explanation of this phenomenon is quite simple. Every market has several limitations, such as the number of potential buyers and sellers as well as the amount of securities for sale. Traders buy securities only if they have optimistic expectations of the future growth. Although this assumption is logical, it causes one interesting paradox related to a whole market. The market is often overbought and tends to turn just when traders' beliefs in a future growth are culminating.[14]

Since markets are multi-level hierarchical systems, a limit cycle at a certain level doesn't represent the relationship between a system and an environment, but it expresses the

[13] The limit cycle is regular, if the corresponding attractor is stable.

[14] This paradox has been proven in dozens of empirical cases, when markets suddenly collapsed without any fundamental reason after a strong growth.

relationship just between two adjacent levels of the market hierarchy. Because of this, behavior of the whole market is described by several nested limit cycles. Limit cycles related to lower levels of the market hierarchy have a shorter time period, narrower oscillation range and they are usually driven by the higher cycles.

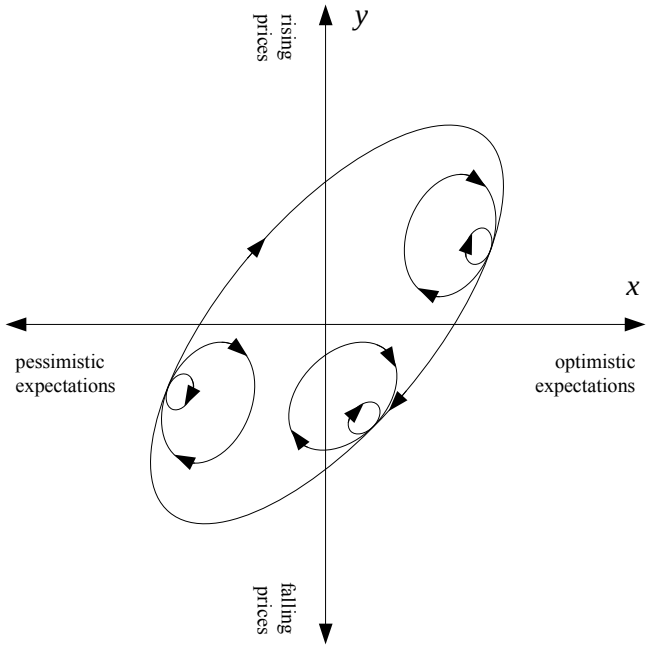


Figure 2.3: The hierarchy of nested limit cycles.

2.1.3 Unexpected events and shocks

Every limit cycle has its own oscillation period, which is synchronized with an upper cycle. The synchronization is performed by information distributed from the upper system to the lower one. If the information occurs in a phase when it is expected, then the limit cycle can adapt to it very quickly, but this is not a major case. Information from the upper system or environment often comes unexpected. Such information is referred as the *shock* and often causes that the crowd must change its dynamic structure to overcome the shock and to synchronize its behavior with the surrounding environment.

In other words, shocks are caused by sudden differences between the actual and the expected prices. According to (2), the shocks usually have two different sources:

- Unexpected and sudden movement of the underlying security prices.
- Unexpected and sudden changes in social, political or economical environment.

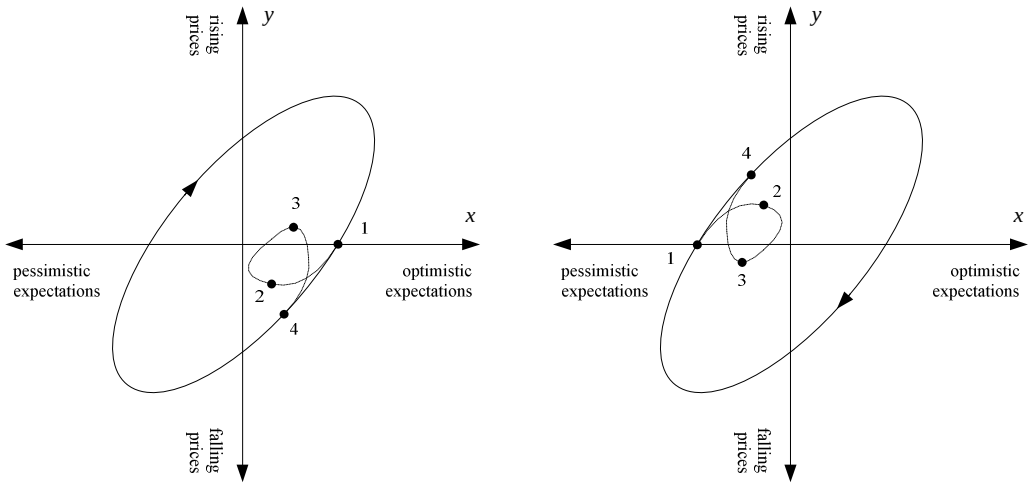
The shocks which overrun traders' expectations in a compliant way are referred as *pro-trend* shocks, while the contradicting ones are referred as *counter-trend*.

The pro-trend shock occurs when market participants suddenly realize that the market prices differ much more than they formerly expected. Pro-trend shocks stimulate integration tendencies of successful crowds which are on the good side of the market movement and

corrupt unity of the unsuccessful ones. Because of this, the pro-trend shocks reinforce strength, stability and duration of market trends.[15]

The counter-trend shock occurs when the market price suddenly and unexpectedly moves against the main trend. Such shock corrupts integration tendencies of the formerly successful crowd and forces traders to reevaluate their attitudes. The counter-trend shocks which are caused by unexpected counter-movements of market prices corrupt crowd's unity much more than fundamental shocks from an environment. Tony Plummer explains the cause of this by using the concept of energy gaps. The shock caused by the unexpected price movement generates an energy gap which cannot be overcome by the crowd itself; meanwhile the shock caused by fundamental information can be easily overcome by the pro-trend energy of a crowd which is not depressed by the fundamental shock.

Shocks caused by unexpected movements of market prices can be also illustrated using the limit cycles. Let's consider the case when the uptrend reaches its top and suddenly changes its direction (see Fig. 2.4.a).



A
B

Figure 2.4: (a) The shock caused by an unexpected depreciation of the previously rising market. **(b)** Unexpected appreciation of the down-trending market also causes a shock.

If the crowd is ready for such change, then the shock doesn't occur and the limit cycle moves from the point 1 directly to the point 4. Otherwise, the shock causes that the crowd rapidly decreases its expectations about the future and the limit cycle moves from the point 1 to the point 2. The rapid depression of traders' optimism causes the temporal depreciation of market prices to a new local minimum represented by the point 2. The market at point 2 is oversold and the prices are inadequately low. There is a minor group of (counter-trend) traders who opened their short positions exactly when market prices reached the top of the cycle. Since these traders don't realize that the main trend is about to change its direction, they usually gain short-term

[15] Integration tendencies of a successful crowd can be illustrated using the example of political parties. When a party is successful and participates on the governance, then the integration tendencies of such party are much stronger than the inner conflicts. Because of this, a governing party rarely falls apart. If a party is unsuccessful in new elections, then the integration tendencies fade out and the formerly irrelevant inner conflicts cause serious downfall.

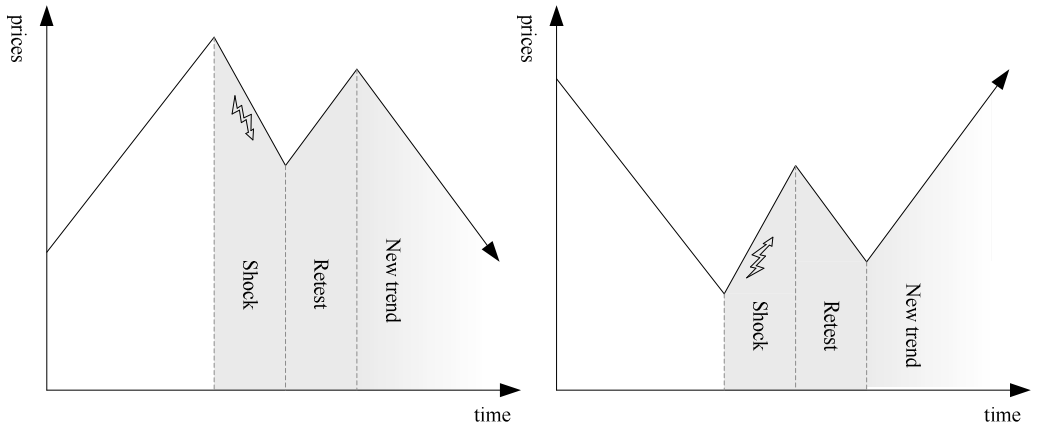
profits and close their positions at the point 2. Because of this, market prices (and also traders' expectations) slightly rises before the great fall (point 3).[16]

This principle is valid also for shocks caused by unexpected appreciations of previously falling markets (Fig. 2.4.b). When the market unexpectedly bounces from the bottom and starts rising, a small group of short-term traders close their long positions (point 2) and pull the prices back to a local bottom. After this, the crowd reorganizes its structure and the new uptrend starts (point 4).

2.1.4 Generalized turnover patterns

The generalized turnover pattern is an idealized scenario for development of absolute prices on markets during an information shock. The previous chapter analyzes market turnovers from the system theory point of view using the limit cycles. Considering that the limit cycles illustrate only changes of market prices, we have to transform them into absolute price levels in order to create patterns which would be applicable to the standard price charts.

We have intuitively sketched three different phases of the market turnover during our analysis of limit cycles. Now, we will define them more precisely (2). Figures 2.5.a and 2.5.b illustrate phases of the top-down and bottom-up turnover patterns.



A | B **Figure 2.5:** The top-down **(a)** and bottom-up **(b)** turnover patterns.

Shock

We have previously defined the information shock as a difference between the expected and actual market prices. When a market is turning over, the expected movement of market prices is denoted by a "1-to-4" curve of the limit cycle in Figure 2.4.a. This is caused by a fact that the crowd as a whole doesn't realize that the market is actually too overbought (or oversold). This lack of information means that markets usually have stronger descent than traders expect. Because of this, the difference between the real (1-to-2) and the expected (1-to-4) movements causes an information shock.

[16] The short rise after the first fall is referred as the *retest* of the previous uptrend.

Retest / Echo

A second phase of the turnover can be understood as the *echo* of the formerly successful crowd. This phase corresponds to the "2-to-3" curve of the limit cycle. As soon as the crowd realizes that an information shock has occurred, it alters its dynamic structure and absorbs the shock. The shock is absorbed mainly by short-term speculators, who opened their counter-positions just when the market reached its peak. The strong counter-movement (1-to-2 curve) motivates these speculators to massively gain their short-term profits. This act recalls a so-called *retest* of the former trend.

New trend

Since the group of short-term speculators which stands behind this retest is only a minor part of the whole crowd, the retest of a former trend doesn't establish a new price maximum. [17] The key aspect in a whole process is a fear. The fear causes that the retest is not a sufficient consolation for the most of previously successful traders. They often close their positions to get rid of the fear, even at the expense of small losses. Because of this, many traders reevaluate their attitudes and open new positions in a direction of the new trend.[18]

In order that the triple-phase turnover is only an idealized model of the real market behavior, the real market movements are often slightly deformed as we can see in Figure 2.6.

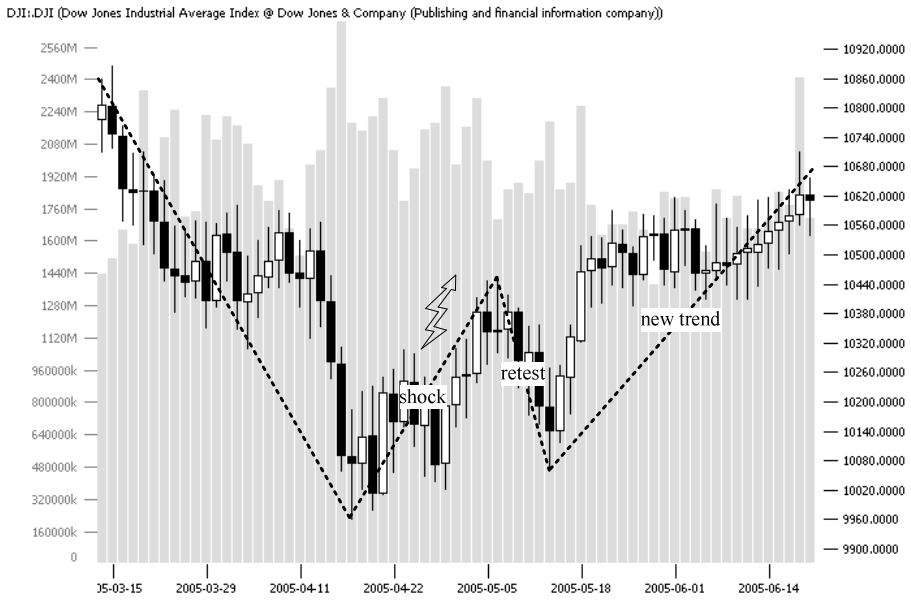


Figure 2.6: The bottom-up turnover in a daily chart of the Dow Jones Industrial Average index.

[17] This section explains generalized turnover patterns especially on examples of the bottom-up turnover. The retest in the top-down turnover involves the establishment of a new price minimum.

[18] This behavior is also known as the *trend-following* strategy.

2.1.5 Generalized pro-trend patterns

In the previous section, we have described the counter-trend shocks and patterns of market turnovers. A counter-trend shock which occurs in the first phase of the market turnover causes an energy gap between the crowd and the surrounding environment. This kind of the energetic gap is sometimes referred as the *first*.

There is also a *second* type of the energetic gap, which often occurs in mature trends. The second energetic gap is caused by an inherent progress of the market trend without any underlying fundamental information. After such progress, the market is implicitly overbought (or oversold) and very unstable (in a relation to the underlying fundaments). Standard behavior of all mature trends is to bounce back from the second energetic gap and to turn the *implicit direction* (ID). This concept is illustrated in Figure 2.7.

Mature trends which reach the second energetic gap by their own inherent progress are very sensitive to positive information shocks from the environment. Because of this, a positive information shock can push the corresponding limit cycle into a completely new oscillation level. The advancement to a new level means not only quantitative but even qualitative changes in the crowd's behavior. This advancement involves a shift of paradigm, structural adaptation and the change of crowd's goals.

However, this is true only in a case when the trend is hard upon the second energetic gap. If a new energy is input into the market before the turning point, it might be rejected or wasted for other purposes than for the spanning of the second energy gap. On the other hand, if this energy enters the market too late after the reversal, it might be not sufficient in a comparison to the power of a new trend.

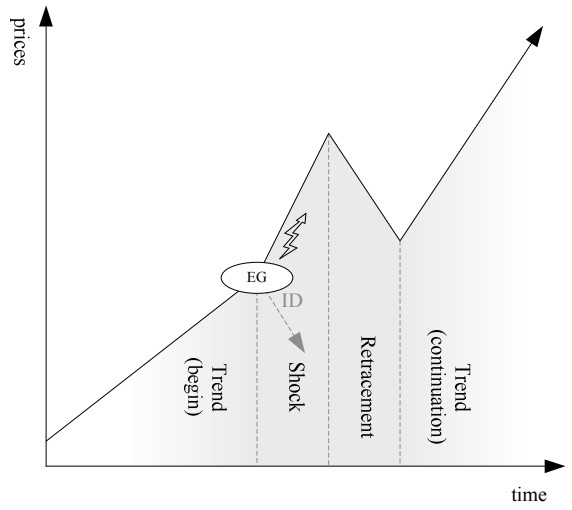


Figure 2.7: When the market trend is about to turn over, a pro-trend information shock can boost it into a completely new level. If no such shock occurs, the market changes its direction in an implicit manner. The dashed arrow (ID) represents the implicit hypothetical direction of the market in a case when no shock occurs.

The positive information shock induces a further movement of the mature trend, which is followed by a retracement. The retracement is an opposite movement of market prices against the main trend. Analogically to the retest phase of a market turnover, retracements are also caused by short-term speculators which gain profits by closing their counter-positions.

Because of this, many market trends must face two different energetic gaps during their lifecycles. The first gap occurs at the beginning of a new trend and it is caused by disproportions between the actual and expected movements of market prices. The second energetic gap often occurs in mature trends as a result of the inherent forces, which push market prices towards new extremes without any prior fundamental reasons. In conclusion, the whole market trend consists of five different phases or waves, as it is shown in Figure 2.9.

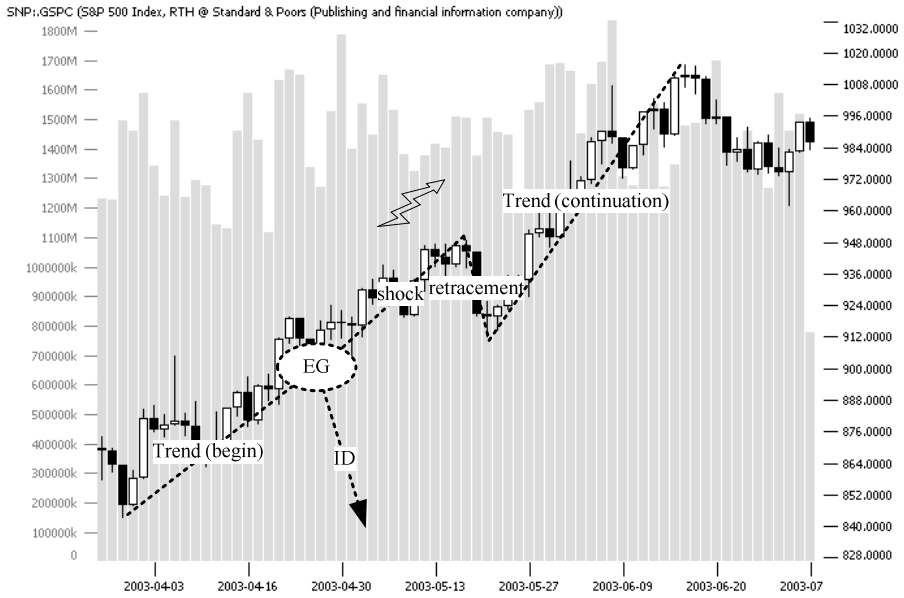


Figure 2.8: The pro-trend information shock in a daily chart of the S&P 500 index.

Since the second energetic gap of the five-wave pattern causes either quantitative or qualitative changes in market trends, the whole pattern involves an evolution. Because of this, the five-wave pattern is a footstone of the *Wave Principle* described by R. N. Elliott in his final major work *Nature's Laws –The Secret of the Universe* in 1946.

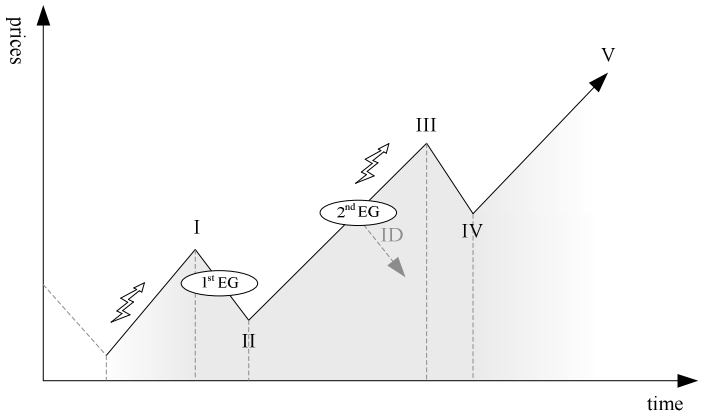


Figure 2.9: The five-wave evolutionary pattern, as it has been proposed in the Elliott's work.

Chapter 3

Basic tenets of automated trading

The first two chapters explained socioeconomic aspects of the financial system. The behavior of crowds is technically covered by the Elliott Wave Principle, which defines several common fractal patterns. Even though the Wave Principle is the most comprehensive analytical tool, it employs also several difficulties. In general, visual analysis and interpretation of charts is a very subjective process, because it lacks hard computing decisions based on hard computing constraints. Ambiguities which occur in charts may cause that traders firstly perform a deep technical analysis and then they make the final decision and final interpretation only on a psychological basis.

Because of this, Flesh-and-blood traders need crisp signals to be objective and to eliminate these psychological aspects. The price chart itself doesn't contain such signals, but we can use derivatives of prices, such as various indicators and oscillators to detect significant events on the market.

The main disadvantage of the automated trading systems is that the output of such systems is very sensitive to parameters of the embedded indicators. A small change in the parameters may lead to the completely different signals which these systems generate.[19]

This chapter introduces readers also to the fundamentals of money management and to the basics of statistical evaluation of trading results. Additionally, we will see that the absolute income is not the most important evaluation criterion of trading systems.

3.1 Indicators and oscillators

One of the problems related to indicators is that every indicator covers different aspect of market behavior. There is no universal indicator which would provide convenient signals in all cases and for all markets. In order that markets employ very dynamic and heterogeneous behavior, every automated trading system sustains successful only for a limited period of time. Because of this, traders must manually choose whether indicator to use. Although the usage of indicators reduces psychological aspects of the trading, the selection of the suitable indicator is still a subjective process.

Generally, we distinguish between several types of indicators according to their purpose. The *trend-following* indicators are used mainly in trending markets. As the title leads, they indicate whether the market has the trending characteristic and where it moves. The examples of the trend-following indicators are moving averages and various directional or convergence / divergence systems.

[19] As we will show later, a soft computing approach and fuzzy logic can be used to eliminate disadvantages of the hard computing trading systems.

Another type of indicators is a group of oscillators which identify turning points in market trends. Because of this, they are mainly used in non-trending markets. Oscillators include Stochastic, Momentum, various RSI indices, Williams %R, Bollinger bands or Bollinger %B, respectively.

According to (14), there also exist *miscellaneous indicators*, which provide insight to the intensity of the market opinion. They involve various ratios (for example the put/call ratio in option markets), commitments of trades and so on. Since miscellaneous indicators are not directly derived from the market prices and they rely on additional data, we won't discuss them in the further text.

3.1.1 Moving averages

The trend-following indicators measure the direction and strength of a market trend. The market is trending, if the corresponding prices have been continuously rising or falling and if there is a big probability that this will be happening also in the near future. We have proposed the Fractal efficiency ratio FER_n in Section 2.1, which measures trending characteristics of markets over the n last bars. The trending markets typically involve big fractal efficiency ratios.

In a nutshell, the basic principle of the trend-following strategy is to identify the trend, to open a position in a proper direction and to close it with a profit when the trend matures. We open positions according to the signals provided by one of the trend-following indicators and close them when the opposite indicator occurs, or when a money management rule forces us to do that.

Simple Moving Average

A *moving average* is a line which represents the smoothed movement of underlying prices. Points of the moving average line are computed for each bar separately as the average of n previous bars:

$$\mathrm{MA}_n(i) = \frac{1}{n}\sum_{j=0}^{n-1} c_{i-j}\,, \tag{3.1}$$

where $\mathrm{MA}_n(i)$ is the i^{th} value of the moving average with the period of n, and c_i is the closing price of the i^{th} bar.

Exponential Moving Average

Additionally, we can use the *exponential moving average* to reflect the short-term memory characteristic of a crowd. The exponential moving average weights closing prices of a chart by numbers from the exponential sequence in order to give more weight to recent prices:

$$\mathrm{EMA}_n(i) = \frac{\sum_{j=0}^{n-1}(1-\alpha)^j \cdot c_{i-j}}{\sum_{j=0}^{n-1}(1-\alpha)^j}, \tag{3.2}$$

where α is a small number within the interval $(0;1)$.

Moving Average Convergence / Divergence

Exponential moving averages are employed also in the *Moving Average Convergence / Divergence* (abbreviated MACD). The MACD involves two lines that correspond to exponential moving averages with different lengths. For example, the shorter average can be computed over the last 12 bars, and the longer one over the last 26 bars. These averages correspond to trends with periods of 12 and 26 bars. The turnover of the shorter trend is indicated by the crossing of two EMA lines and the MACD indicator is then computed as a difference between these averages (Eq. 3.3).

$$\mathrm{MACD}_{\mathrm{n,m}}(i) = \mathrm{EMA}_n(i) - \mathrm{EMA}_m(i) \tag{3.3}$$

The MACD indicator also involves the MACD *signal line* which is computed as an exponential moving average of the MACD indicator (Eq. 3.4).

$$\mathrm{MACDsignal}_{\mathrm{m,n,o}}(i) = \mathrm{EMA}_o(i) \text{ of } \left[\mathrm{MACD}_{m,n}(j)\right]_{j=0}^{o-1}, \tag{3.4}$$

where $\left[x_j\right]_{j=0}^{n-1}$ is a sequence of numbers $x_0, x_1, \ldots, x_{n-1}$.

The relation between the MACD and MACD signal line is represented by the MACD *histogram*, which is computed as a difference between these two lines (Eq. 3.5).

$$\mathrm{MACDhistogram}_{\mathrm{m,n,o}}(\mathrm{i}) = \mathrm{MACD}_{m,n}(i) - \mathrm{MACDsignal}_{\mathrm{m,n,o}}(i) \tag{3.5}$$

The MACD histogram is an ultimate oscillator which indicates the turnover of a trend. A new uptrend starts when the MACD histogram crosses the zero level from the bottom. Similarly, a new downtrend is detected when the histogram crosses the zero level from the top. This concept is illustrated in Figure 3.1.

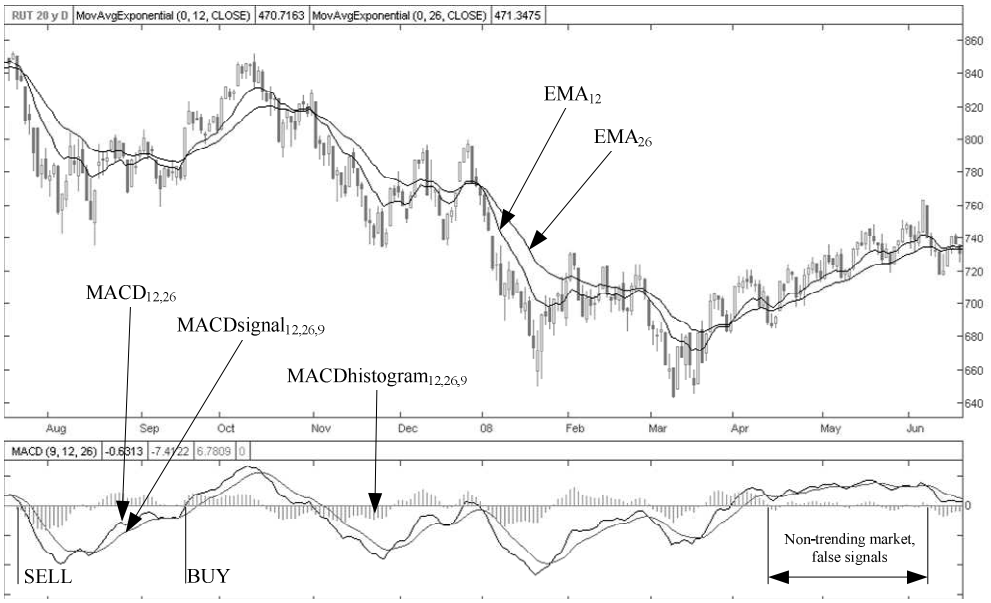


Figure 3.1: The example of the MACD indicator derived from prices of the Russell 2000 index between year 2007 and 2008.

As we have mentioned in the previous text, one of the problems related to indicators is a fact that they are very sensitive to their parameters. This is true also for the MACD. Figure 3.1 illustrates the MACD signaling system for a set of parameters (12, 26, 9), but there is no doubt that different parameters would produce completely different signals. Because of this, many robust trend-following systems combine several moving averages together, such as the *Moving Ribbon* indicator illustrated in Figure 3.2.

Figure 3.2: The example of the Moving Ribbon indicator derived from prices of the Russell 2000 index between year 2007 and 2008.

3.1.2 Average Directional Index

The Average Directional Index is also considered to be a trend-following method. It has been proposed by J. W. Wilder. The directional system identifies how much the price range of the current bar differs from the previous bar's price range. The directional system is constructed as follows.

At first, we identify the *Directional Movement* by comparing today's price range to the yesterday's one. The constellation of consequent candlestick bars implies to the four types of directional movements (see Eq. 3.6 and Fig. 3.3).

$$\mathrm{DM}(i)=\begin{cases} h_i - h_{i-1} & \text{if} \quad h_i > h_{i-1} \wedge l_i > l_{i-1} \\ l_{i-1} - l_i & \text{if} \quad l_i < l_{i-1} \wedge hi < h_{i-1} \\ 0 & \text{if} \quad h_i - h_{i-1} = l_{i-1} - l_i \\ \max\{h_i - h_{i-1}; l_{i-1} - l_i\} & \text{if} \quad h_i - h_{i-1} \neq l_{i-1} - l_i \end{cases} \tag{3.6}$$

In the next step, we compute the true range between the current and previous bar, which is given as a maximum value of distances between the current high and current low, current high and previous close or current low and previous close, respectively (Eq. 3.7).

$$\mathrm{TR}(i) = \max\{|h_i - l_i|, |h_i - c_{i-1}|, |l_i - c_{i-1}|\} \tag{3.7}$$

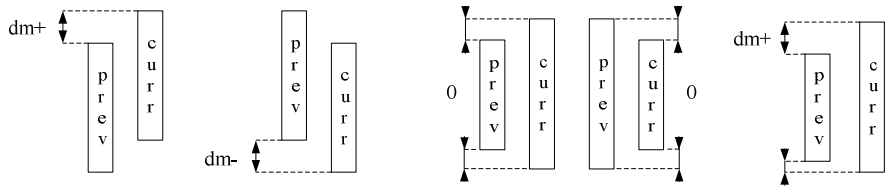


Figure 3.3: The four types of directional movements according to the arrangement of the *current* and *previous* candlesticks.

The third step is based on a calculation of daily directional indicators DI^{+} and DI^{-}. They express directional movements as percentages of the today's true range (Eq. 3.8).

$$\mathrm{DI}^{+}(\mathrm{i}) = \frac{\mathrm{DM}^{+}(\mathrm{i})}{\mathrm{TR}(\mathrm{i})} \;;\; \mathrm{DI}^{-}(\mathrm{i}) = \frac{\mathrm{DM}^{-}(\mathrm{i})}{\mathrm{TR}(\mathrm{i})} \tag{3.8}$$

We compute moving averages with a period of thirteen separately for positive and negative directional indicators (DI_{13}^{+} and DI_{13}^{-}). Finally, these averages are used for a calculation of the Average Directional Index (Eq. 3.9).

$$\mathrm{ADX}(i) = \mathrm{EMA}_{13}(i) \text{ of } \left[\frac{\mathrm{DI}_{13}^{+}(i-j) - \mathrm{DI}_{13}^{-}(i-j)}{\mathrm{DI}_{13}^{+}(i-j) + \mathrm{DI}_{13}^{-}(i-j)} \right]_{j=0}^{13-1} \tag{3.9}$$

As we have mentioned earlier, the Average Directional Index doesn't indicate direction of the trend, but only the strength. This index has a range between the 0.0 and 1.0, or between 0% and 100%, respectively. Values below 0.2 indicate weak trends and values above 0.4 indicate strong trends.

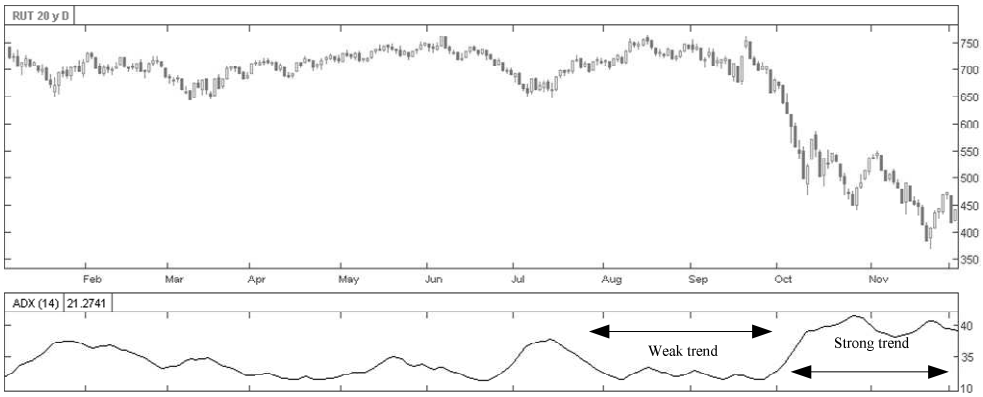


Figure 3.4: The Average Directional Index applied to a daily chart of the Russell 2000.

3.1.3 Average True Range

The Average True Range is another indicator proposed by J. W. Wilder. It measures the volatility of a market price movement. The Average True Range is computed as an exponential moving average of true ranges of the n previous bars (Eq. 3.10).

$$\mathrm{ATR_n}(i) = \mathrm{EMA}_n \text{ of } \left[\mathrm{TR}(i-j)\right]_{i=0}^{n-1}, \tag{3.10}$$

where $\mathrm{TR}(i)$ is a true range of the i[th] bar, as it has been proposed in Equation 3.7.

Figure 3.5 illustrates Average True Range on the example of the Russell 2000 index in comparison with the previously introduced ADX indicator. Although both indicators are based on the true ranges of individual bars, Average True Range doesn't regard the trend as a whole (in contrary to the ADX indicator and Fractal Efficiency Ratio).

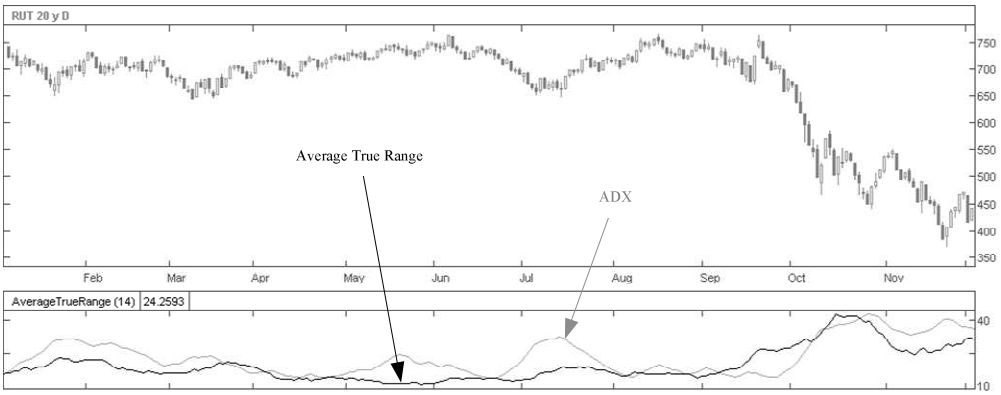


Figure 3.5: The comparison of the Average True Range and Average Directional Index.

3.1.4 Relative Strength Index

The Relative Strength Index (RSI) is an indicator of the market's short-term balance. It indicates whether a market is overbought or oversold. If the market is overbought, then it means that such market contains an extraordinary and unreasonable big amount of long positions. Similarly, the oversold market is characterized by a big amount of short positions. In other words, the balance of the market is corrupted in a case, when the prices don't correspond to a structural organization of the underlying crowd.

From the technical analysis point of view, the market is considered to be oversold or overbought if the following two premises are satisfied. At first, market prices are pushed to new local maximums or minimums, respectively. In order that this happens in a short period of time, such movements are often extraordinary strong. Secondly, the disrupted balance is often indicated by increased short-term volatility.

The relative strength index (RSI) is computed as follows. At first, for each bar in a sequence of n bars, the *upward* and *downward* changes are calculated. If the current closing price is higher than the previous one, then the upward change is given as a difference between these two prices. The downward change is calculated in a similar manner for the falling market,

where the current closing price is lower than the previous one. Please note that both upward and downward changes are positive numbers and the upward change is zero for falling markets as well as the downward change is zero for rising markets (Eq. 3.11).

$$\mathrm{UC}(i)=\begin{cases} c_i - c_{i-1} & \text{if} \quad c_i > c_{i-1} \\ 0 & \text{if} \quad c_i \le c_{i-1} \end{cases},\ \mathrm{DC}(i)=\begin{cases} c_{i-1} - c_i & \text{if} \quad c_{i-1} > c_i \\ 0 & \text{if} \quad c_{i-1} \le c_i \end{cases} \tag{3.11}$$

The Relative Strength Index is then computed using the exponential moving averages of upward and downward changes over the sequence of n last bars (Eq. 3.12).

$$\mathrm{RSI}_n(i)=1-\frac{1}{1+\dfrac{\mathrm{EMA}_n(i)\ \text{of}\ \left[\mathrm{UC}(i-j)\right]_{j=0}^{n-1}}{\mathrm{EMA}_n(i)\ \text{of}\ \left[\mathrm{DC}(i-j)\right]_{j=0}^{n-1}}} \tag{3.12}$$

It is recommended to use exponential moving averages with a period of fourteen. The RSI index oscillates in a range between 0.0 and 1.0. Values below 0.3 (30%) indicate oversold markets and values above 0.7 (70%) indicate that a market is overbought.

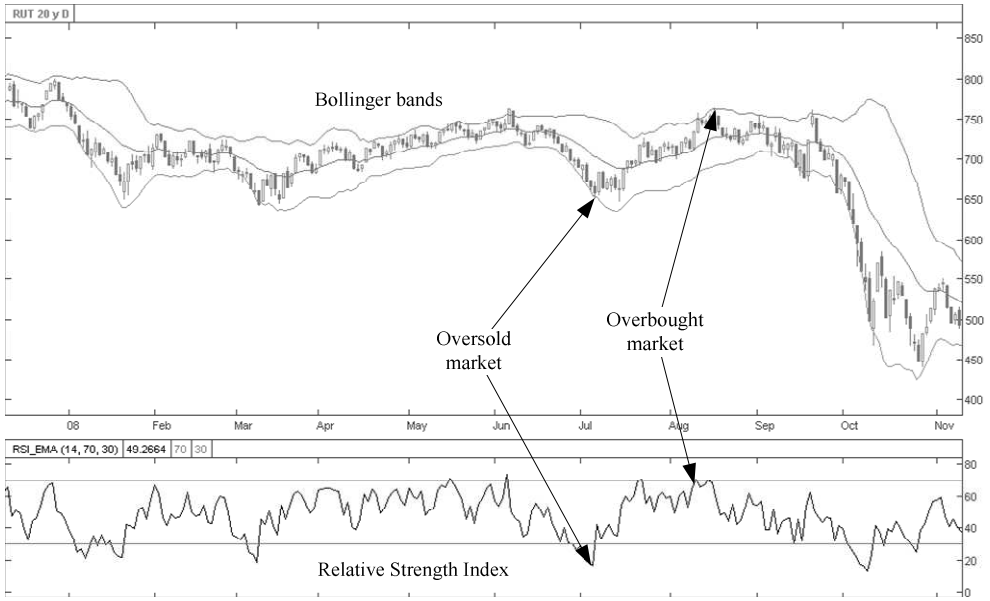


Figure 3.6: Bollinger Bands and Relative Strength Index are indicators of the short-term market balance.

3.1.5 Bollinger Bands

Bollinger Bands is another indicator of market's short-term balance. Similarly to the RSI index, the Bollinger Bands also indicates whether the market is overbought or oversold. This indicator consists of three lines (bands) which follow the basic price movement. The middle band is a simple *n*-period moving average. The upper and lower bands are deviated from the middle one by a *k*-multiplied standard deviation of n last closing prices (see Eq. 3.13).

$$\begin{aligned} \mathrm{MB}_n(i) &= \mathrm{SMA}_n(i) \\ \mathrm{UB}_{n,k}(i) &= \mathrm{MB}_n(i) + k \cdot \mathrm{std}\left[c_{i-j}\right]_{j=0}^{n-1}, \\ \mathrm{LB}_{n,k}(i) &= \mathrm{MB}_n(i) - k \cdot \mathrm{std}\left[c_{i-j}\right]_{j=0}^{n-1} \end{aligned} \tag{3.13}$$

where $\mathrm{std}\left[x_j\right]_{j=0}^{n-1}$ is a standard deviation of the sequence of numbers $x_0, x_1, \ldots, x_{n-1}$.

A typical length of the moving average (*n*) is 20 and typical multiplier of the standard deviation (*k*) is 2 (see Fig. 3.6).

A market is considered to be overbought, if it closes above the upper Bollinger band. Similarly, the oversold market reaches the bottom Bollinger band.

Bollinger %B

There also exists a mechanical oscillator derived from the Bollinger bands. It is often referred as the Bollinger %B and tells where the prices are in a relation to the Bollinger Bands. The Bollinger %B is defined by Equation 3.14.

$$\%\mathrm{B}_{n,k}(i) = \frac{c_i - \mathrm{LB}_{n,k}(i)}{\mathrm{UB}_{n,k}(i) - \mathrm{LB}_{n,k}(i)} \tag{3.14}$$

Values above 1.0 indicate that the prices are above the upper band (overbought market). Values below 0.0 represent oversold markets and prices below the bottom band.

3.2 Money management

Every mechanical trading system is based on two pylons. The first one is a signaling system based on indicators and oscillators. A signaling system identifies valuable trading opportunities, but it doesn't provide advices whether a concrete trader should utilize a certain opportunity according to the size of his capital and according to other subjective parameters.

Because of this, the signaling system is focused on objective market events, whereas the subjective trader's situation and financial possibilities are covered by the second pylon of a mechanical trading system – the *money management*. The money management also provides an answer whether a specific market opportunity is appropriate, considering the financial resources of a concrete trader.

Every operation on the financial market employs two basic parameters – the risk and the reward. These parameters have continual proportion, which means that trader willing a bigger reward must accept also a bigger risk. The main goal of the money management is to maximize rewards while protecting the capital against the bankruptcy.

The risk

The classical definition of a risk is based on an assumption that the risk is determined by a standard deviation of the expected outcome, but money management uses another definition. We rather define risk as a worst-case outcome of the trade. As we will see later, traders are able to control the risk using the concept of stop-losses. The stop-loss is a mental barrier that

determines worst acceptable conditions of the trade. If a trade goes bad, the trader must close the position in order to get rid of the mental stress.

An extremely important thing is that the stop-losses designated to protect wealth cannot be moved in an adverse way. In fact, moving the stop-loss in an adverse way is just like buying the hope for real cash. The stop-loss, as a mental concept, corresponds to a conditional stop order, which is triggered by reaching the stop price (as a mental barrier).

There is an indirect proportion between a probability of the stop-loss execution and the size of the corresponding loss. Stop orders which are too close to the current market price cause many small losses, whereas placing stop orders in a bigger distance generates fewer bigger losses.

The reward

When the market moves in a friendly direction, successful traders may close their position with a profit. A predetermined price level at which a trader closes the successful trade is referred as the *profit target*. Similarly to stop-losses, profit targets are also mental barriers which must be determined strictly by the money management rules.

The indirect proportion exists also between a size of the profit and a probability that the profit will be reached. When we choose a profit target which is close to the current market price, we can expect a small profit with a big probability and vice versa for further profit targets. Profit targets are usually implemented by limit orders.

Risk-reward ratio

A stop-loss in the combination with a profit target brings two possible outcomes of the trade. In a better case, the trade is closed at the profit target with a positive reward, while the worse case means the stop-loss execution. Both stop-loss and profit target have a certain probability that they will be filled. The ratio between how much money we risk and how much we are able to earn is referred as the *risk-reward ratio* (Eq. 3.15).

$$\mathrm{RRR} = \frac{p_{pt} - p_o}{p_o - p_{sl}}, \tag{3.15}$$

where p_o is a price at which the position is opened, p_{pt} is a limit price of the profit target order and p_{sl} is a stop price of the stop order.

The bigger risk-reward ratio between stop-loss and profit target orders means a bigger chance that such position will be closed with a loss. Similarly, the lower risk-reward ratio is associated with a lower, but more probable reward. The dependency between the risk-reward ratio and the accuracy of a signaling system (SSA) is given by Equation 3.16.

$$\mathrm{SSA} = \frac{1}{\mathrm{RRR} + 1} \tag{3.16}$$

Let's consider the following example (Fig. 3.7). We enter the long position on the 9th of April 2008 by buying 100 shares of NYSE:BA[20] for $77.39 each. We limit the maximum possible loss to $361 by placing the stop order at $73.78. If the trade will move in an appropriate

[20] Common shares of the Boeing, inc. listed at New York Stock Exchange.

direction, we close it at the predefined profit target $83.78 with a total profit of $639. These orders must be embedded in the following First-triggers-OCO bracket:

```
# 1ST TRIGGERS OCO BRACKET START
BUY +1000 BA @77.39
SEL -1000 BA STP 73.78 GTC
SEL -1000 BA @83.78 LMT GTC
# BRACKET END
```

The risk-reward ratio of such trade is then given as:

$$\mathrm{RRR} = \frac{p_{pt} - p_o}{p_o - p_{sl}} = \frac{83.78 - 77.39}{77.39 - 73.78} = 1.77 \tag{3.17}$$

Considering the given risk-reward ratio, the accuracy of the signaling system must be at least the following:

$$\mathrm{SSA} = \frac{1}{\mathrm{RRR} + 1} = \frac{1}{1.77 + 1} = 0.36\,, \tag{3.18}$$

which means that at least 36% of all trades must successfully reach the profit target level.

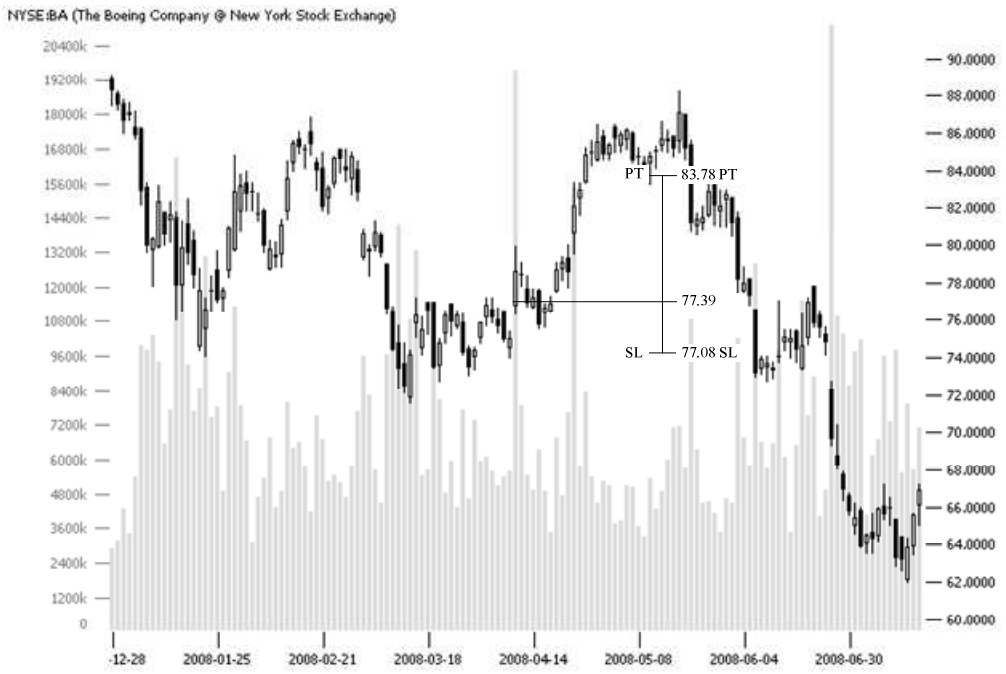


Figure 3.7: The risk-reward ratio is determined by the placement of profit targets and stop-losses.

Please note that the accuracy of a signaling system means that the system is able to generate signals, which lead to the execution of profit targets. Because of this, this accuracy has an implicit dependence on the distance of profit targets.

Another approach is to use moving stop-losses instead of fixed profit targets. The concept of moving stop-losses protects accumulated wealth and simultaneously keeps the future growth unlimited. Because of this, moving stop-losses are often used on trending markets with big

fractal efficiency ratios (FER_n). Similarly, fixed profit targets and fixed stop-losses are more suitable on markets which involve chaotic behavior.

The following table concludes several aspects of the trading on different markets according to their fractal efficiency ratios:

Market	Non-trending	Trending
FER_n ratio	Small	Big
Time horizon	Short	Long
Recommended money management strategy	• Profit target + stop-loss	• Moving stop-loss • Trailing stop

3.3 Statistics

The speculative trading is intended not to find a good long-standing investment, but to systematically and iteratively raise capital by a big number of short-standing trades. In contrary to investments, the short-term speculations are diversified not to different contracts, but to different trades. Because of this, the statistical evaluation of trading reports is a very important topic in construction of automatic trading systems.

We are able to statistically evaluate whether the outcome of a trading system on specified data is statistically meaningful, or if it is only a product of the curve-fitting[21]. Generally, most of statistical analyzers are able to determine the following aspects of the simulation of trading system:

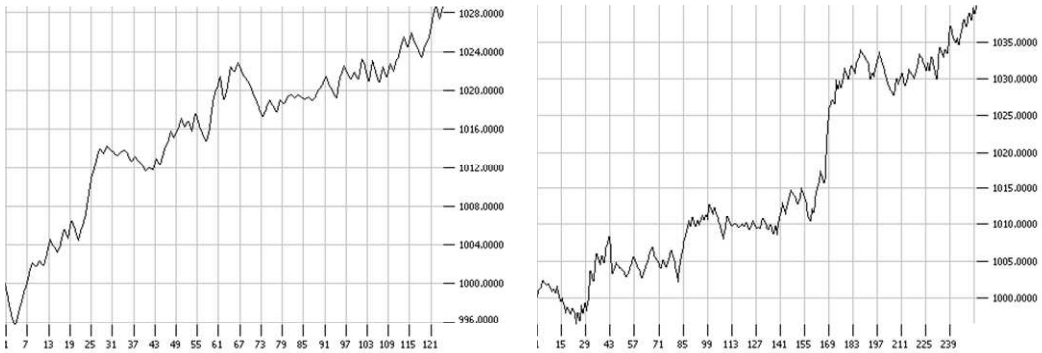
- Number of trades, wins and losses
- Number of profit target exits
- Number of stop-loss exits
- Number of exits caused by a signaling system
- Maximum and average number of stop-loss moves
- Maximum, minimum and average number of contracts per trade (position sizing)
- Highest, lowest and final amount of equity
- Percentage return on starting equity
- Largest and average profit or loss
- Average and maximum profit or adverse excursion
- Biggest drawdown of the trading account
- Maximum, minimum and average duration of position
- Annual return (calculated using the compound interest)

The statistics above is computed from the trade report, which contains the following information about each trade:

[21] Curve-fitting is the process of estimating parameters and coefficients of a trading system in order to optimize it on a given set of data. Such system has a great profitability regarding the in-sample data and known market conditions, but it lacks robustness and stability needed for the real trading and out-of-sample data.

- Entry and exit timestamps for each position
- Ticker of the security
- Quantity
- Entry and exit price
- Transaction fees
- Cumulative profit or loss

The overall performance of a trading system is often graphically illustrated using a so-called *equity curve*. This curve shows how the trader's equity grows over the time. Consider the following two equity curves in figures 3.8.a and 3.8.b.



A | B **Figure 3.8:** Equity curves of two different strategies applied on the same market data (er2u7 futures).

These curves demonstrate outcomes of two different trading strategies applied on the same market data (er2u7 futures[22]). Both strategies begin with a starting capital of $1000. As we can see, the second strategy reaches the higher final equity ($1039.90) than the first one ($1028.52). Because of this, if we consider only the absolute outcome, the second strategy seems to be better. However, the first strategy has the following advantages:

- Smaller number of trades
- Better ratio between the numbers of wins and losses
- Smaller maximum drawdown of the trading account
- More regular equity curve

All of these characteristics mean that the first strategy is better, although it has the smaller final outcome. For example, fewer trades generate smaller transaction costs (which are not considered in this example).

When we take a look at the second equity curve (Fig. 3.8.b), we see that most of the profit is caused by one extraordinary successful trade. Is the outcome of such trade the representative sample of a whole trading system? We don't think so.

More regular equity curve and the smaller account drawdown means that the behavior of the first strategy is less risky. Because of this, such strategy may involve bigger leverage compensating the smaller absolute outcome.

[22] Intraday E-mini Russell 2000 Sept 2007 futures market (er2u7) between 2007-09-05 and 2007-09-07.

The following table contains complete statistics of both strategies. The statistics of the first strategy corresponds to the first equity curve in Figure 3.8.a, whereas the second strategy represents the second equity curve in Figure 3.8.b.

First strategy		Second strategy	
Start	2007-09-05 16:00:04	Start	2007-09-05 00:28:00
End	2007-09-07 21:43:29	End	2007-09-07 21:53:30
Number of trades	124	Number of trades	254
Number of wins	64	Number of wins	119
Number of losses	60	Number of losses	135
Number of profit target exits	64	Number of profit target exits	120
Number of stop-loss exits	60	Number of stop-loss exits	134
Number of strategy exits	0	Number of strategy exits	0
Average number of stop-loss moves	0.00	Average number of stop-loss moves	0.00
Max contracts per trade	1	Max contracts per trade	1
Min contracts per trade	1	Min contracts per trade	1
Avg contracts per trade	1.00	Avg contracts per trade	1.00
Equity start	1000.00	Equity start	1000.00
Equity high	1028.59	Equity high	1039.90
Equity low	995.75	Equity low	996.38
Final equity	1028.52	Final equity	1039.90
Return of starting equity	2.85%	Return of starting equity	3.99%
Largest win	2.84	Largest win	5.56
Largest loss	-2.35	Largest loss	-3.05
Average win	1.21	Average win	1.19
Average loss	-0.82	Average loss	-0.75
Max consecutive wins	6	Max consecutive wins	6
Max consecutive losses	7	Max consecutive losses	8
Average profit excursion	0.81	Average profit excursion	0.72
Average adverse excursion	-0.57	Average adverse excursion	-0.54
Max profit excursion	2.84	Max profit excursion	5.56
Max adverse excursion	-2.35	Max adverse excursion	-3.05
Max drawdown	5.69	Max drawdown	6.32
Max position duration	4 hrs 37 min	Max position duration	12 hrs 22 min
Min position duration	19 sec 98 ms	Min position duration	15 sec 42 ms
Avg position duration	16 min 3 sec	Avg position duration	16 min 6 sec

3.4 The sensitivity to changes of parameters

The previous section introduced readers to the statistical evaluation of trading strategies and illustrated it on two examples, but it didn't reveal the strategies themselves. This section analyzes one trend following strategy and shows how the parameters of such strategy affect the overall outcome. The sensitivity of the trading system to changes of parameters is the major problem of all hard computing systems.

Similarly to the previous section, we will trade the E-mini Russell 2000 futures with the expiration month of September 2007. The timeframe of the data is one minute. The strategy is based purely on a simple moving average signaling system. Since our objective is to demonstrate sensitivity of the signaling system, we don't use any money management methods.

The simulation is started at the market opening on 1st of September 2007 and continues until the 15th of the same month. The strategy is very simple and it is constructed as follows (see Code 3.1).

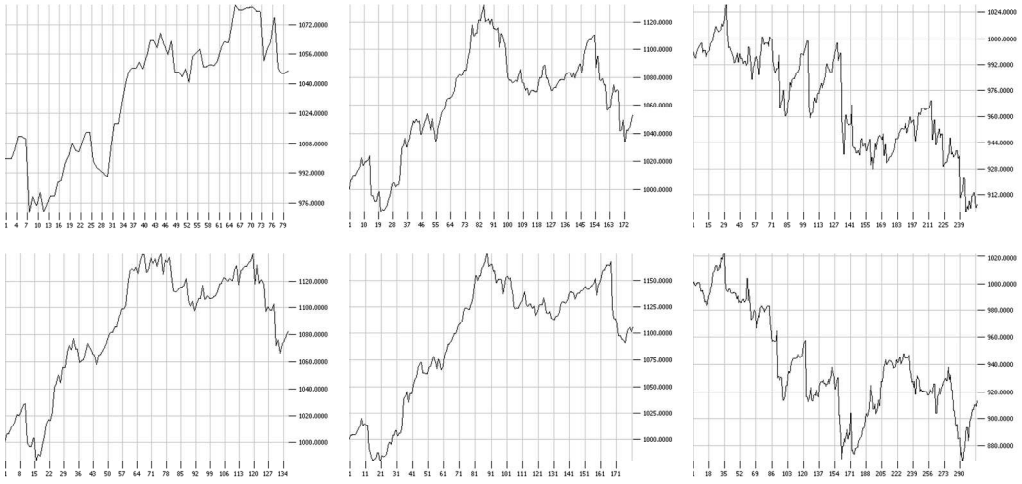
Code 3.1:

```
if SMA_n crosses SMA_m from the bottom then
begin
   if out_of_position then BUY +5 er2u7
   else SEL -5 er2u7
end
else if SMA_n crosses SMA_m from the top then
begin
   if out_of_position then SEL -5 er2u7
   else BUY +5 er2u7
end
```

There are two parameters - m and n. The first parameter stands for a period of the shorter moving average, whereas the second one stands for a period of the longer moving average. The strategy is referred as trend following because it changes the position every time when a new trend is detected by the crossover indicator. Figures 3.9.a-f contain equity curves for the following parameters:

case	A	B	C	D	E	F
m	52	32	25	23	16	13
n	25	9	7	11	9	6

As we can see, the overall outcome and behavior of this strategy is very sensitive to changes of the parameters. In fact, such strategy is very unstable and its performance depends also on the market conditions. Because of this, we cannot expect similar results for different conditions.



A	C	E
B	D	F

Figure 3.9: Equity curves of the same strategy with different parameters. As we can see, the bigger parameters of the moving average cover the bigger trends on the market.

Chapter 4

Simulation and backtesting of trading strategies

The previous chapter introduced readers basic tenets of the automated trading and proposed several entry and exit strategies based on a wide spectrum of various indicators and oscillators. Although it explains how trading strategies generate recommendations and how to statistically evaluate the outcome of such strategies, it doesn't make readers familiar with the principles of such evaluations.

This chapter discusses several aspects of the simulation of trading environment and proposes several models of the time and price skews that occur on markets. It introduces readers to the basics of the Discrete Event System Specification and constructs an appropriate trading simulator. In other words, this chapter provides some technical knowledge and resources needed for the simulation of trading strategies which have been proposed in the previous chapter.

4.1 The value of simulation in the trading

There are two ways how to figure out the successfulness of an unverified strategy. The first one is to implement the strategy into an automatic trading system and to run such system on the real market.[23] The disadvantages of this approach are quite clear. At first, the trader risks his own capital and usually doesn't have enough time and financial resources to gain real practical experiences with such strategy. Secondly, the real trading doesn't provide opportunities to comparatively experiment with either the different parameters of the same strategy or with the completely different strategies.

The theory of modeling and simulation has a practical application in many scientific and technical domains. Experiments with real objects and processes are often too expensive, time-consuming, dangerous or unsuitable. In such cases, the real processes are abstracted into the corresponding computational models, and the experiments are performed on these models rather than on real objects or processes.

Considering the trading point of view, the impact of automated trading strategies can be studied on the model of trading environment based on historical data. This approach is often referred as the *paper trading*. The simulated trading lacks most of the disadvantages of the real trading. Besides the capital and time savings, it allows traders to repeatedly test different strategies on the same market conditions. The modeling and simulation allows traders to perform what-if analyses on the parameters and comparative tasks on different strategies.

As we will see later, the simulation of trading strategies can be used also for the optimization tasks.

[23] There are many commercial trading platforms which implement ATS functionalities, such as the TradeStation™, Strategy Runner™ or others.

4.2 Human factor in the trading chain

Generally, trading systems may represent traders in a trading environment in two different ways. The indirect representation means that the trading system provides advises to a trader, who manually reflects them into the trading orders. Such trading systems are often referred as *semi-automatic*, because they involve a human factor in the trading chain.

If a broker provides an appropriate application program interface, then the trading system can be connected directly to the broker. In such cases, the trading system enters orders into the market autonomously without any trader's intervention. Because of this, the systems directly connected to the broker's API are referred as *automatic*.

Figure 4.1 illustrates data chains of the automatic and semi-automatic trading systems. The human factor in a semi-automatic trading system improves its robustness and eliminates many errors caused by the AI's reasoning.

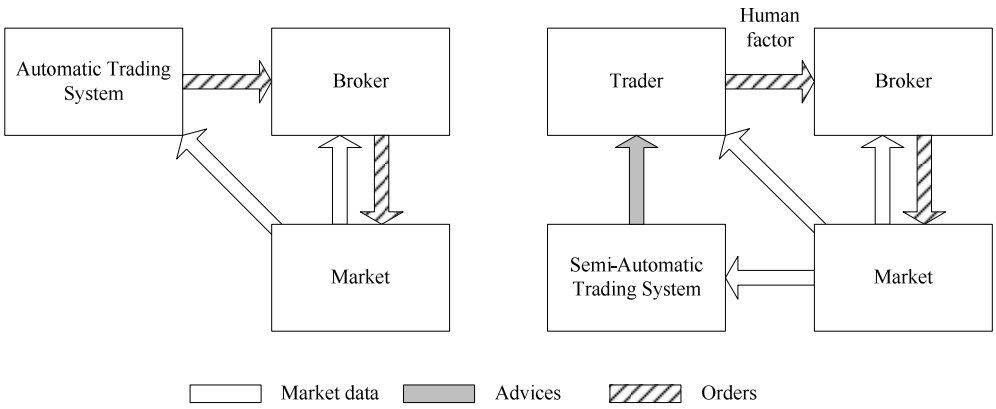


Figure 4.1: Data chains of the automatic and semi-automatic trading systems.

From the simulation point of view, the human factor involved in semi-automatic trading systems causes that the simulation of these systems is inaccurate, especially in a combination with the data based on a short timeframe. Because of this, we will consider in the further text only fully automated trading systems.

4.3 Modelling of the intra-bar price movements

The integration of information and communication technologies into financial markets as well as the increased volume of trades caused that market prices are changing very rapidly. In very liquid markets, the prices are changing within seconds. Because of this, a trader (or a trading system, respectively) must react to these changes immediately.

Even though the abstract feelings on a market move in a continuous manner, exchange clearing systems provide a discrete stream of quotes as new bids are paired together. This stream is characterized by an ordered sequence of quotes $[Q_i]_{i=0}^{n-1}$, where the quote Q defined as a duple:

$$Q = (t, p), \tag{4.1}$$

where p is the price quoted at the corresponding time t.

The stream of prices is sampled into so-called *bars* for the purposes of recording, backtesting and further analysis. Each bar is associated with a disjunctive time period and statistically covers underlying movement of prices. The most common type of a bar is the *OHLC*[24] or the *candlestick* bar. An OHLC bar carries opening, closing, highest and lowest price which occurred within the underlying time period. As we can see in Figure 4.2, the candlestick bar has the same meaning, but different graphical representation than the OHLC bar. Since the OHLC provides only a statistical summary of what happened during a concrete time period, we must model prices at every single moment of this period according to the statistical summary of such bar. Let's consider a OHLC bar, which covers a stream of quotes $\left[Q_i\right]_{i=0}^{n-1}$. The statistical characteristics are then computed as follows (Eq. 4.2):

$$\begin{aligned} o &= p\left(Q_0\right) \quad h = \max\left[Q_i\right]_{i=0}^{n-1} \\ c &= p\left(Q_{n-1}\right) \quad l = \min\left[Q_i\right]_{i=0}^{n-1} \end{aligned}, \tag{4.2}$$

where n is the number of quotes covered by the bar.

There are several ways how to model underlying price movements according to the OHLC characteristics. We have proposed a simple model which interpolates prices within the time period of a bar in the following way. Equation 4.3 interpolates rising bars (where $o < c$), but analogical equations can be constructed also for falling bars. This interpolation is graphically illustrated in Figure 4.2.

$$p\left(t\right) = \begin{cases} o + \dfrac{\left(h-o\right)\cdot\left(t-t_o\right)}{\frac{1}{2}\left(t_c-t_o\right)} & \text{if} \quad t \le \tfrac{1}{2}\left(t_o+t_c\right) \\ c + \dfrac{\left(c-l\right)\cdot\left(t-t_c\right)}{\frac{1}{2}\left(t_c-t_o\right)} & \text{if} \quad t > \tfrac{1}{2}\left(t_o+t_c\right) \end{cases}, \tag{4.3}$$

where o, h, l and c are statistical characteristics of the bar, $\left\langle t_o, t_c \right\rangle$ is a time period covered by the bar and $p\left(t\right)$ is the price valid in a certain time t.

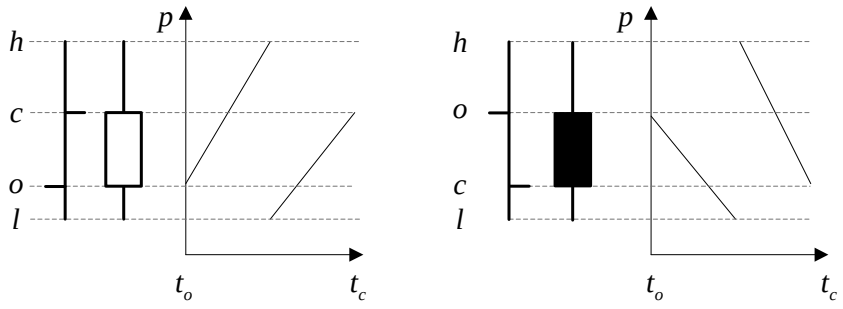


Figure 4.2: The interpolation of prices between the opening and closing moment of the bar.

[24] OHLC is an abbreviation for the Open/High/Low/Close statistical characteristics of the bar.

4.4 Modelling of the order execution

In connection with the interpolation of intra-bar prices, there is also a need to model the execution of various types of orders. Considering the given order, we must model when and at what price this order will be filled. Historical data with a shorter timeframe provides us more precise information about the prices in a certain time. Longer timeframes bring uncertainty to price movements and decrease the quality of the simulation. This problem is notable especially on daily charts. Because of this, the execution of orders must be modeled with a respect to the model of intra-bar price movements proposed in Section 4.3.

Let's assume the common price function $p(t)$, which models development of a price in a certain time moment t. This section proposes several execution models for market, limit and stop orders.

Market orders

Considering that market orders don't carry any additional fulfillment conditions, they are immediately executed at the market price. Because of this, the execution time t_{exe} is modeled as a time t_{now} when an order has been posted plus small transaction skew t_{Δ}. The corresponding price is then given as $p(t_{now} + t_{\Delta})$.

Limit orders

In contrary to market orders, a limit order carries the limit price, which represents the worst-case price of its fulfillment. It is clear that the limit price defines an upper limit for the buy orders and a bottom limit for the sell orders. Limit orders wait in a queue until the following conditions are satisfied:

- For the buy orders, the low l_i of an actual bar is smaller than the limit price.
- For the sell orders, the high h_i of an actual bar is higher than the limit price.

If an order meets the conditions above, it may be executed within the actual bar. The question is just when and at what price. The execution of limit orders is illustrated in Figure 4.3.

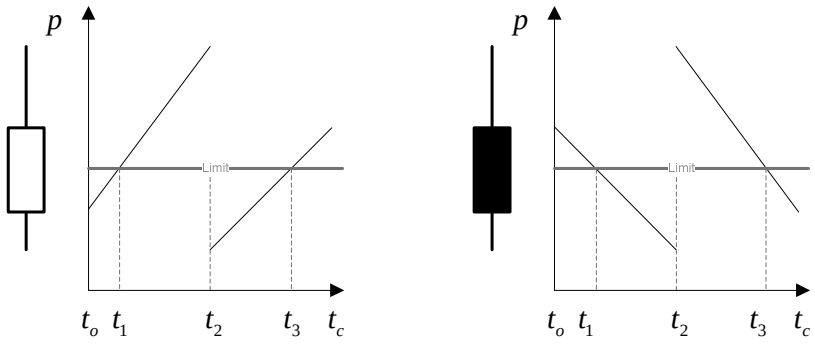


Figure 4.3: The significant time intervals which determine whether the order is or is not executable. These intervals are defined by the limit price and by the order side.

The determination of the significant time points t_1, t_2 and t_3 is based on a calculation of the corresponding equations (Eq. 4.3). These points divide the time domain into the intervals in which an order is or is not executable.

For buy orders, the time domain at which these orders are executable is determined by the interval $I_{buy} = \langle t_{now} + t_\Delta; \infty \rangle \cap \left(\langle t_0; t_1 \rangle \cup \langle t_2; t_3 \rangle \right)$. For sell orders, the executable interval is given as $I_{sell} = \langle t_{now} + t_\Delta; \infty \rangle \cap \left(\langle t_1; t_2 \rangle \cup \langle t_3; t_c \rangle \right)$, where t_{now} is the time where an order is posted and t_Δ is a transaction skew. In normal cases, orders will be executed at the time $t_{exe} = \min(I)$ with the corresponding price $p(t_{exe})$.

Stop orders

The stop order is a specific type of an order used for the purposes of money management. It has associated a stop price. The stop order is executed when the market price reaches the level predefined by a stop price. Stop orders are modeled in a similar manner than limit orders. Considering Figure 4.3, stop orders are executable only in the time points t_1 and t_3. These points correspond to a time domain determined by the interval $I = \langle t_{now} + t_\Delta; \infty \rangle \cap (t_1 \cup t_3)$. Similarly to the previous case, a stop order will be filled at the time $t_{exe} = \min(I)$ with the price $p(t_{exe})$.

4.5 Modelling of the time and price skews

The communication chain between a trader, broker and a security exchange involves various types of the time and price skews. Price skews in volatile markets significantly affect the profitability of a whole trading strategy. Because of this, we have to model these skews in order to achieve better correspondence with the reality.

Generally, we distinguish between the time and price skews. The time skews are caused by communication latencies in the trading chain, while the price skews are represented by a difference between the officially quoted price and the price at which the order is actually executed. The price skews are included mainly in market orders.

Communication-based time skews

The source of time skews is a relatively complicated process of order fulfillments. The length of a skew depends on a way how orders are posted between the trader and broker as well as between the broker and the security exchange. Earlier, when orders have been posted via telephone lines and manually paired by pit brokers, the tenths of minutes have been spent during the execution.

Nowadays, the fully automated electronic communication systems lead to an almost immediate execution of posted orders. For purposes of the simulation, we model only skews between the trader and broker using the Gaussian (normal) distribution. We assume that the transaction delay is characterized by the Gaussian distribution with parameters $\mu = 20s$ and $\sigma = 10s$. This skew takes importance mainly in volatile intra-day markets with short timeframes. However, such skew doesn't matter in the position trading.

Liquidity-based time skews

The other important source of the time skews is a phenomenon that occurs in less liquid markets – the absence of counterparty. Every posted order waits in a queue until someone posts the opposite order. Since many clearing algorithms don't allow the partial execution of orders, bigger transactions must wait until the clearing machine gathers the corresponding amount of opposite orders. Because of this, the time skew of orders depends also on a volume and it is modeled as follows (Eq. 4.4):

$$t_{\Delta} = N \cdot \frac{t_c - t_o}{V_i}, \tag{4.4}$$

where t_{Δ} is the time skew caused by the absence of counterparty, N is a volume of the order (the amount of securities to trade), V_i is the total volume traded on the market within a certain time period (typically one bar) and $t_c - t_o$ is a length of such period (typically a difference between the opening and closing time).

Price skews

As we have mentioned earlier, the price at which a certain order will be filled can differ from the quoted market price. This is caused by the principle, which the clearing algorithms operate on. Considering that the quality of the order execution depends also on the volume of such order, we have proposed several models of price skews.

The *volatility slippage model* skews the price by a percentage of the current bar's true range:

$$p_{\Delta} = c \cdot \mathrm{TR}(i), \tag{4.5}$$

where $\mathrm{TR}(i)$ is the true range of the current bar and c is a percentage constant randomly generated using the Gaussian distribution $\mu = 0$ and $\sigma = 0.5$

The second model is the *tick slippage model* which skews the price by a random multiple of the price tick. The tick is a smallest step by which the price can move on the market.

4.6 Simulation of the trading environment

The simulation model is based on a modified version of the coupled DEVS model, which is interpreted by a modified DEVS simulator. Although the fundamental principles introduced in Section 4.6 remain unchanged, several minor customizations have been made in the simulator to meet requirements of the trading model.

Specifics of the model

The purpose of the model is to approximate a trading environment. The trading model consists of the *Automated Trading System* (abbreviated *ATS*), *Order Execution* and the *Data Provider*. The trading environment involves various types of time and price skews, as they have been

proposed in Section 4.5. The communication flow employs orders, cancels, acknowledgements and price quotes. Figure 4.4 illustrates basic arrangement of the model components.

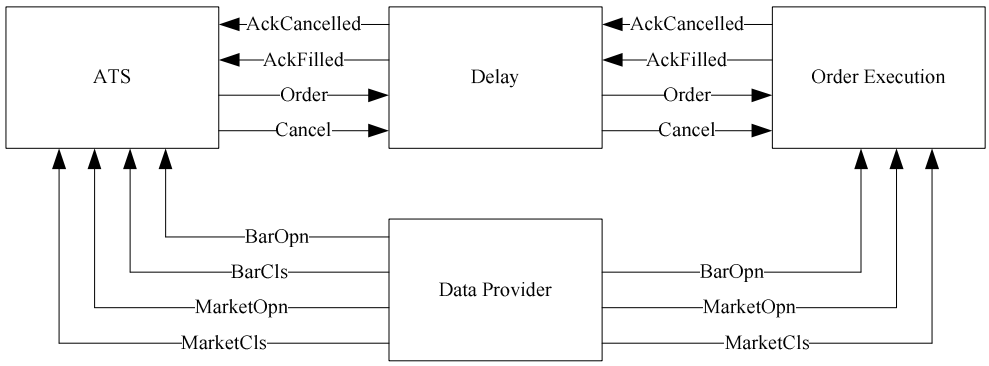


Figure 4.4: Arrangement of the trading model components

As we can see, the interconnection of components doesn't respect the standard information flow between a trader, broker and a market.

In the real environment, a broker acts only as an intermediary between traders and a market. This means that it forwards all orders and cancels from traders to the market and acknowledgements back from the market to traders. The execution of orders is normally performed by a security exchange.

Considering the purpose of our model, the trading environment has been altered as follows. The model is focused mainly to a trader and *ATS*, respectively. Because of this, the communication chain between traders, brokers and the security exchange is generalized only to a delayed interaction between the *ATS* and the *Order Execution* components.

Secondly, orders are not forwarded to the security exchange by the broker, but the *Order Execution* component virtually simulates their fillings. Hence, the *Data Provider* only provides a stream of quotes to the *ATS* and the *Order Execution* components. This ensures that both these components have always the same information about the market situation.

Specifics of the simulator

As we have mentioned earlier, trading simulator contains several differences from the standard DEVS formalism. In contrary to DEVS, messages in our trading model are structured and parameterized. For example, the *Order* message covers market, limit and stop orders. As we will see later, the *Order Execution* component employs different behavior for different orders. This fact is in a contradiction with the external transition function δ_e, which is given as $\delta_e : Q \times X \to S$.

The external transition function involves also another customization. In a conventional DEVS model, an external transition between states of the atomic model doesn't generate any output. Our model is allowed to do so. Even though this problem can be solved in a standard DEVS by using two individual transitions with one auxiliary state, we have modified the implementation of the simulator.

Each state in the atomic DEVS usually has at most one implicit transition statically determined by the internal transition function $\delta_i : S \to S$. In our model, implicit transitions are planned dynamically according to the input data. It is important to realize that one algorithmically defined atomic component can be formally described by several different DEVS

models. Our model determines the new state by a function of the old state and the input data. Other models can treat the same state with different data as completely different states. Such approach makes the model compatible with a formal definition of the atomic DEVS, but it makes it unnecessarily complicated.

The time period after which an atomic component implicitly moves to a new state is not determined by a simple function of such state ($ta(s)$). The function ta in our trading model is defined algorithmically and depends also on input data and other conditions.

Essentially, the DEVS formalism defines the total state Q as a duple $Q=(s,e)$, where s is the state in a time context e. According to previously described modifications, we formally define the total state of components in our trading model as a triple $Q=(s,e,r)$, where r is an additional parameter which represents input data and content of various components' registers.

4.6.1 The Data Provider component

The *Data Provider* component is an abstraction of the functionality provided by a security exchange or a broker. It has only one purpose, to provide a stream of price quotes. In the reality, the security exchange provides real-time quotations of the prices asynchronously from any time frames, and it is a customer's interest to react to them as quickly as possible. The only problem is that we don't have price quotes available for every moment during the simulation. Section 4.4 introduces function that approximates intra-bar price movements. There are two solutions regarding such approximation.

The first solution is to sample the bar into several sufficiently short intra-bar points. Then, we compute the price for every such point and generate the corresponding quote. For example, we can sample a bar with the time frame of one minute into the sixty individual intra-bar points in order to emulate the stream of quotes with a period of one second. This solution corresponds to a type of simulation which is sometimes referred as the *activity scanning*. The activity scanning is quite fast, but not very precise approach.

The second solution is to send each bar at once in the time, when it is closed. If we consider a chart based on the hourly time frame, the automated trading system typically receives such bar at the closing bell. This would be enough for a long-time position strategy, but it is not enough for a precise modeling of the order execution. Because of this, the *Order Execution* component usually receives bars at the corresponding opening times. The *Data Provider* component generates two events for each bar, one event at the opening moment for the *Order Execution* component, and one event at the closing moment for the *ATS* component. The behavior of the *Data Provider* component is illustrated in Figure 4.5.

The *Data Provider* alternates between the `BAR_OPEN` and the `BAR_CLOSE` state. In the `BAR_OPEN` state, the component generates the corresponding OHLC quote for the *Order Execution* component and notifies the *ATS* component about the opening price (but not about the whole OHLC characteristics). In the `BAR_CLOSE` state, the component generates a full OHLC quote also for the *ATS*. The *Data Provider* can read historical data from the file, database or the data warehouse.

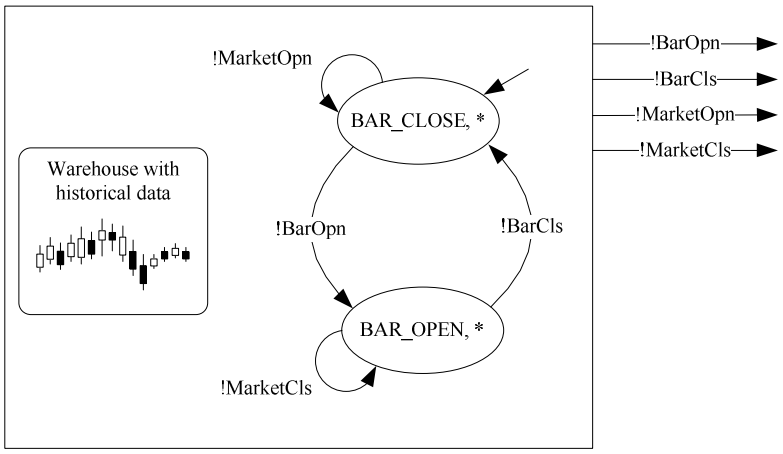


Figure 4.5: The *Data Provider* component of the coupled trading model. This component oscillates between two states and generates a stream of price quotes.

4.6.2 The Delay component

The *Delay* component doesn't implement traditional functionality of the atomic DEVS model. As the name suggests, it is used mainly to delay orders, cancels, acknowledges and other messages that passes through it. The *Delay* component implements a model of communication and liquidity-based time skews, as they have been proposed in Section 4.5.

Despite the fact that this component is conceptually modeled as atomic, it implements its own calendar and several other functionalities of a DEVS coordinator. However, it cannot be treated as a state machine, because it has only one state.

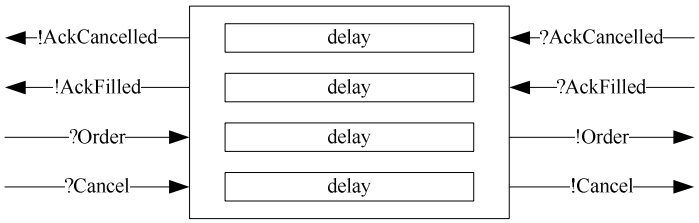


Figure 4.6: The *Delay* component propagates all signals from input to output ports with a predefined delay.

Every time when a new order arrives, the *Delay* component determines the communication and liquidity-based latency and plans a new output event in the time $t' = t + t_\Delta$, where t is the time where the order was received and t_Δ is the modeled latency. Due to this latency, an output event cannot be generated simply by calling the *externalMessage* method of the influenced component, but it must be planned in a calendar similarly as in the implementation of the coupled DEVS model.

4.6.3 The Order Execution component

The *Order Execution* component simulates execution of orders entered by the automated trading system. Similarly to the *Data Provider*, the *Order Execution* is also only an abstraction that covers functionality provided by a security exchange and by a broker. Besides order execution, real brokers usually manage traders' accounts, provide financial leverages and other services. In contrary to these conventions, this component simulates only the execution of clients' orders and doesn't tend if a certain client has appropriate financial resources needed for the transaction.

The *Order Execution* combines standard behavior of an atomic DEVS and the behavior which cannot be modeled using the DEVS specification. In addition, this component contains also a queue of waiting orders and the register with the actual price bar (Figure 4.7).

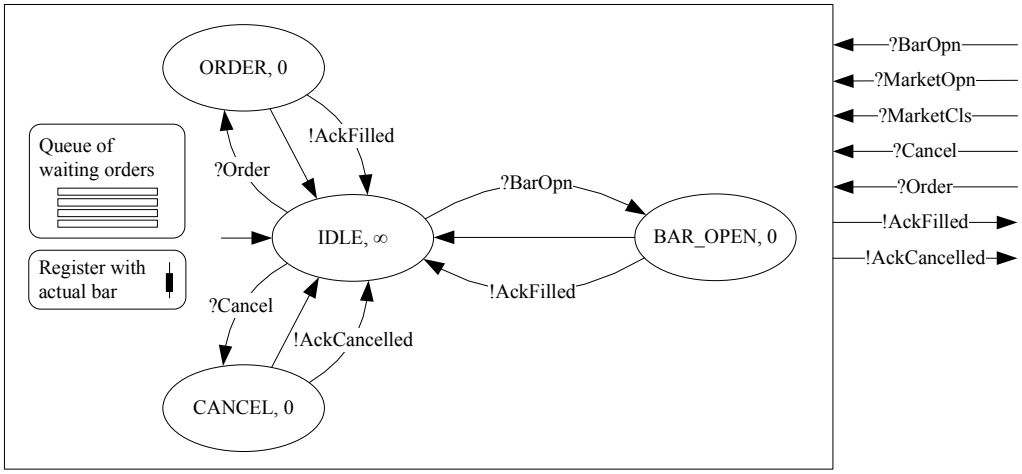


Figure 4.7: The *Order Execution* component of the coupled trading model simulates execution of orders.

The *Order Execution* periodically receives OHLC quotes and corresponding opening times from the *Data Provider*. When a new quote is received via the `BarOpn` message, this component moves into the `BAR_OPEN` state and stores the received OHLC quote into the *Actual Bar Register*. Immediately after that, the *Order Execution* component moves back to the `IDLE` state and tries to fill all executable orders from the waiting queue.

Considering the actual price bar, a given order is executable if it meets requirements defined in Section 4.4. The simulator determines time and price aspects at which the order will be filled and plans the corresponding output messages (`AckFilled`). Since the output messages are planed generally in an arbitrary time asynchronously to the state of the automaton, the *Order Execution* component doesn't generate output events using the *externalMessage* method, but it has its own calendar (similarly to the *Delay* component).

When a new order is received, the component moves to the `ORDER` state and compares the order to the actual price bar. If the order is executable within this bar, the component plans a new `AckFilled` message according to the model of order execution introduced in Section 4.4. Otherwise, if the order is not executable within the actual bar, the component puts it into the waiting queue and returns to the `IDLE` state.

When the component receives a `Cancel` message, it turns into the `CANCEL` state, removes the corresponding order from the queue, generates a new `AckCancelled` message and returns

back to the IDLE state. If the cancel is misled, the machine turns into the IDLE state without any additional actions.

4.6.4 The ATS component

The *ATS* (Automated Trading System) component is the most sophisticated component in a whole simulation model (see Figure 4.8). It implements behavior of a trader or an automated trading system, respectively. This component respects the main concern of all speculative traders, which is to buy a security in order to sell it at a higher price (and vice versa for short trades). The *ATS* component implements a mechanical trading strategy, limited data window, money management, statistics, position controller and a mechanism for tracking orders.

The limited data window contains n last bars, which are actualized as new quotes are received from the *Data Provider* component. The embedded trading strategy analyzes this window to decide whether to enter or exit the position. The size of the quote window depends on a used strategy. For example, if the strategy is based on exponential moving averages with periods of 12 and 21, then the data window must be at least 21 bars long.

The *ATS* component waits in the IDLE state until a new quote from the *Data Provider* is received. If this happens, the component actualizes its data window and moves into the DECISION_I state. The internal transition function from the DECISION_I state implements an intelligent behavior, which decides whether to enter the position by posting SE, PT and SL[25] orders and whether to return back into the IDLE state and wait for better market conditions.

When the *ATS* decides to enter a position, it posts one SE order. If the embedded strategy involves also money management, it may additionally post the PT and SL orders. After that, the automaton moves into the PENDING_I state and waits for proper acknowledges from the *Order Execution* component. Please note that the time spent in the PENDING_I state depends also on a type of used order. The market SE orders are usually executed very quickly, whereas conditional entries realized by limit orders may cause that the *ATS* will be waiting excessively long in the pending state. Current implementation of the *ATS* automaton doesn't allow the strategy to cancel pending entry orders in the PENDING_I state.

In a POSITION state, the trading system holds securities. When a new bar is received, the *ATS* actualizes the data window and turns into the DECISION_P state which implements mechanisms that decide whether or not to exit the position. When the strategy decides to exit, it generates an exit order and cancels all pending conditional orders (profit targets and stop-losses). After that, the *ATS* component turns into the PENDING_P state.

The automaton remains in the PENDING_P state until the *ATS* component receives a positive acknowledgement of the exit order (AckFilled) and acknowledgements of all cancels (AckCancelled). If the component doesn't register any further pending orders, the automaton moves into the IDLE state and logs position into the trade report.

Generally, the position can be closed in two different ways. The first way is the previously described strategy exit, when the *ATS* generates an exit order. The position can be closed also by conditional PT and SL orders in terms of the money management. Stop-loss and profit target orders close the position asynchronously to the trader's behavior. If this happens, the automaton moves from the POSITION state into the PTSL_CATCH state and logs position

[25] SE, SX, PT and SL are abbreviations for the strategy enter, strategy exit, profit target and stop-loss orders.

statistics into the trade report. After that, the *ATS* cancels all pending orders and waits for acknowledgements in the `PENDING_P` state. If there are no further pending orders, the automaton returns back into the `IDLE` state and the whole process can repeat.

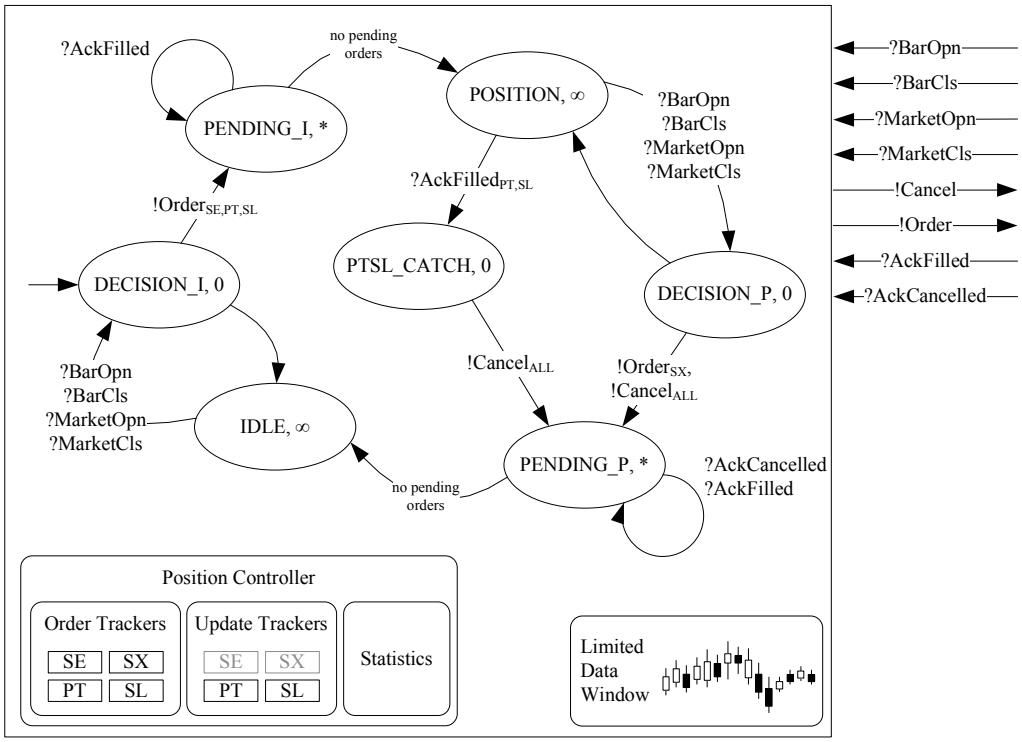


Figure 4.8: The *ATS* component of the coupled trading model interprets automated trading strategies, manages trading positions and tracks orders.

Position controller and order trackers

The non-detachable facility of the *ATS* component is a *position controller*. As the name suggests, the position controller controls and maintains the actual opened position. It records all its properties, such as:

- Entry and exit times
- Profit and adverse excursions
- Side of the position and quantity of securities
- Blocked margin

Orders can be posted in four different roles – the strategy entry, strategy exit, profit target and the stop loss. The execution of these orders must be tracked by the *ATS* in order to ensure that no order will be lost. For example, the position won't be regarded as closed until all profit targets and stop losses will be cancelled.

The position controller uses *order trackers* for all roles of orders. There are two mandatory order trackers – the strategy entry (SE) and the strategy exit (SX). Additionally, conditional orders required by money management are tracked by the profit target (PT) and stop-loss (SL) trackers.

Each order tracker defines five states, in which the order can be (see Fig. 4.9). The `EMPTY` state means that the corresponding order tracker doesn't contain an order. If an order is present in the tracker, but it hasn't been posted to an exchange, it is regarded as `NOT_FIRED`. The tracker is in the `PENDING` state, if it contains an order which has been actually posted to the exchange, but not filled yet. When the *ATS* component receives a positive acknowledgement (`AckFilled`) of a certain order, it turns the corresponding order tracker into the `FILLED` state. Similarly, the negative acknowledgement (`AckCancelled`) of a certain order turns the tracker into the `CANCELLED` state. An order can be cancelled only if the corresponding order tracker is in the `PENDING` state.

When the embedded strategy wants to post an order to the exchange, it just places such order into the order tracker. The tracker itself ensures that the order will be properly posted, executed and acknowledged.

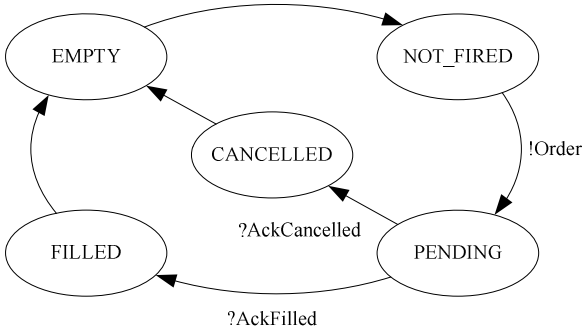


Figure 4.9: The order tracker defines five states, in which the order can be. Even though arrows illustrate possible transitions between states in a relation to the output and input events, this diagram as a whole doesn't represent the DEVS automaton.

4.6.5 The parallel run of multiple trades

Traders may hold several different positions and trade them independently. For example, the intraday trading of the corn futures can be combined with the position trading on forex markets. Positions can vary in a size, leverage or in a time horizon. The only common element is the trading account. Because of this, both profits and margins are accounted together, regardless positions which they correspond to.

Figure 4.4 illustrates the relation between components *ATS*, *Delay*, *Data Provider* and *Order Execution*. This relation forms an environment for trading of one type of security.

The parallel run of multiple trades can be achieved by the creation of multiple *ATS* components. Considering that the *Data Provider* is connected to the *ATS* component as well as the *Order Execution* component is connected (via *Delay*) to the *ATS*, this principle involves several routing difficulties. This approach requires a router which would be able to route orders between the *Delay* and *ATS* components according to their tickers (see Figure 4.10).

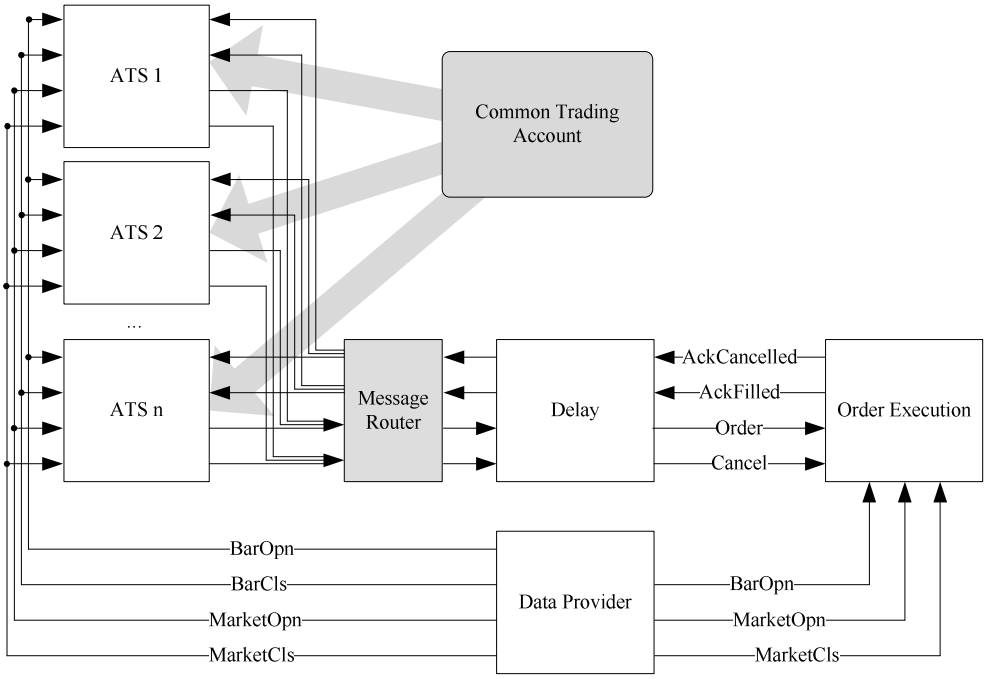


Figure 4.10: An environment for the parallel run of multiple trades. This approach requires the message router to handle messages from various ATS components.

The second approach uses multiple independent tetrads of components proposed in Figure 4.11. Each tetrad consists of the detached *Data Provider*, *Order Execution*, *Delay* and *ATS* components. The only common feature is the trading account. This approach doesn't require the message router, but it produces many futile components performing tasks which can be easily done by the common one.

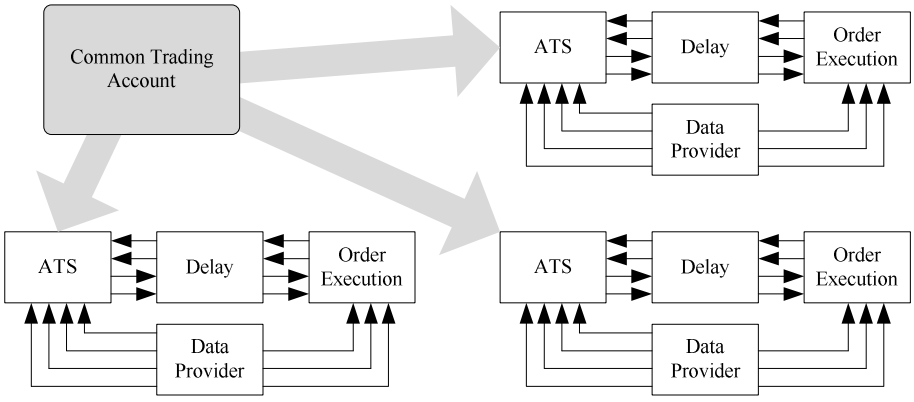


Figure 4.11: Another approach of the trading environment for the parallel run of multiple trades. This approach doesn't require message router, but it produces many futile components.

4.7 Embedding trading strategies into the simulation

Section 4.6.4 explains how the *ATS* automaton moves between the states according to a corresponding position on the market. The automaton performs intelligent decisions in the states `DECISION_I` and `DECISION_P`. The `DECISION_I` represents a state in which the *ATS* is out of the position (idle) and decides whether to enter it. Similarly, the `DECISION_P` state implements an intelligent behavior that decides whether to exit the position.

The *ATS* component embeds a trading strategy that implements several methods. These methods analyze situation on the market and generate appropriate trading decisions. Considering the object oriented paradigm, the trading strategy is a class that implements the interface shown in Code 4.1.

Code 4.1:

```
interface StrategyInterface
begin
   methods:
      getWindowCapacity()
      outOfPositionBarOpened(t, o)
      outOfPositionBarClosed(t)
      inPositionBarOpened(t, o)
      inPositionBarClosed(t)
      positionEntered(t)
      outOfPositionMarketOpened(t, o)
      outOfPositionMarketIsAboutToClose(t)
      inPositionMarketOpened(t, o)
      inPositionMarketIsAboutToClose(t)
end
```

The `getWindowCapacity` method returns a minimum required size of the limited data window. The *ATS* component calls this method during its initialization.

The `outOfPosition*` methods are called in cases when the ATS component is out of the position. Because of this, these methods typically implement entry strategies and involve mechanisms which decide whether to enter the position.

Similarly, the `inPosition*` methods implement behavior of the ATS component in cases when it is in the position. Since this, these methods correspond to the exit strategies.

Methods are distinguished also by the type of event occurred. The `*BarOpened` methods are called every time when a new `BarOpn` message arrives. A `BarOpn` message notifies the *ATS* about the opening price of the corresponding bar. Analogically, the `*BarClosed` methods handle `BarCls` messages. In contrary to the `BarOpn` messages, the `BarCls` messages carry the whole OHLC characteristics.

There also exist market opening and closing notifications, which are usable mainly for intraday trading strategies. The ATS receives a notification every time when the market is opened. The message that carries such notification is denoted as `MarketOpn` and the handling methods are named as `*MarketOpened`.

The other useful notification is the `MarketCls`. It is generated every time when the market is about to close. The corresponding `*MarketIsAboutToClose` methods usually react by closing of intraday positions.

4.7.1 Simulation case study

Previous sections theoretically explained basic simulation principles and construction of the simulation model. This section employs these principles in a practical case study. The core of the study is a simulation model that embeds a simple trend-following strategy. The strategy is run on the historical intra-day data. This study shows how *ATS* takes decisions according to the input data and how messages are handled by the components in this model.

We will use the same data (er2u7[26] futures) as we have used in Section 3.4. The simulation starts at the market opening on the 1st of September 2007 and continues until the 15th of the same month. The object-oriented implementation of the strategy looks as shown in Code 4.2.

Code 4.2:

```
class MyStrategy extends Strategy implements StrategyInterface
begin
   constants:
      m = 32          // Period of the longer SMA
      n = 9           // Period of the shorter SMA
      q = 100         // Quantity of traded securities
      sl = 2          // Size of stop-loss (in a ratio to the ATR of 10 bars)
   methods:
      method getWindowCapacity()
      begin
         return m
      end
      method outOfPositionBarClosed(t)
      begin
         let x = MA_m(i) − MA_n(i)
         let x' = MA_m(i−1) − MA_n(i−1)
         // Determines if and how the shorter SMA crosses the longer one
         if x > 0 ∧ x' < 0 then
               SE → addOrder("BUY + q er2u7")
         else if x < 0 ∧ x' > 0 then
               SE → addOrder("SEL - q er2u7")
      end
      method inPositionBarClosed(t)
      begin
         // Moves the stop-loss in a positive way if the "unprotected" profit exceeds the
         // predefined value diff .
         let atr = ATR_10(i)
         let diff = atr · sl
         let sl_old = SL → getOrder() → getStopPrice()
         if positionController → side = long ∧ c_i − sl_old > diff then
```

[26] Intraday E-mini Russell 2000 Sept 2007 futures.

```
            SL ⟶ replaceOrder("SEL -q er2u7 STP c_i − diff GTC")
        if positionController ⟶ side = short ∧ sl_old − c_i > diff then
            SL ⟶ replaceOrder("BUY +q er2u7 STP c_i + diff GTC")
    end
    method positionEntered(t)
    begin
        let atr = ATR_10(i)
        let p_SE = SE ⟶ getOrder() ⟶ getFillPrice()
        if positionController ⟶ side = long then
            SL ⟶ addOrder("SEL -q er2u7 STP p_SE − atr · sl GTC")
        else
            SL ⟶ addOrder("BUY +q er2u7 STP p_SE + atr · sl GTC")
        end
    end
end
```

The strategy takes entry decision according to the mutual position of two moving averages. The strategy generates a long-position entry signal when the shorter moving average crosses the longer one from the bottom. Similarly, the short-position entry signal is generated just when the shorter one crosses the longer one from the top. This behavior is implemented in the `outOfPositionBarClosed` method.

Positions are closed exclusively by the money management rules. When the *ATS* receives the acknowledgement of the entry order (with the known entry price), it calculates a corresponding stop-loss level and posts a stop order. If the market moves in an appropriate direction, the *ATS* periodically moves the stop-loss in order to protect the accumulated profit. This behavior is implemented in the `inPositionBarClosed` method.

The simulation of this strategy on the intraday data of er2u7 futures produces the following stream of events (Code 4.3).

Code 4.3:

```
Date        Time      Comp./State    Event          Content
----------  --------  -------------  -------------  ------------------------------
2007/09/02  23:51:00  ATS/IDLE       ?BarCls        er2u7 793.6 794.0 793.6 794.0
2007/09/02  23:52:00  ATS/IDLE       ?BarCls        er2u7 794.4 794.4 793.9 793.9
...
2007/09/03  11:55:00  ATS/IDLE       ?BarCls        er2u7 794.0 794.0;794.0 794.0
(ATS decides to enter a short position)
2007/09/03  11:55:00  OE/IDLE        ?Order         SEL -100 er2u7
2007/09/03  11:55:04  ATS/PENDING_I  ?AckFilled     SEL -100 er2u7 (fa 793.7)
(First position entered at 793.7, adding first stop-loss ...)
2007/09/03  11:55:04  OE/IDLE        ?Order         BUY +100 er2u7 STP 794.0 GTC
2007/09/03  11:56:00  ATS/POSITION   ?BarCls        er2u7 793.7 793.7 793.7 793.7
2007/09/03  12:08:00  ATS/POSITION   ?BarCls        er2u7 793.8 793.8 793.8 793.8
...
2007/09/03  12:32:00  ATS/POSITION   ?BarCls        er2u7 793.8 793.8 793.2 793.2
(Adding a new stop-loss in order to secure the accumulated profit ...)
2007/09/03  12:32:00  OE/IDLE        ?Cancel        BUY +100 er2u7 STP 794.0
2007/09/03  12:32:00  OE/IDLE        ?Order         BUY +100 er2u7 STP 793.6 GTC
2007/09/03  12:32:01  ATS/POSITION   ?AckCancelled  BUY +100 er2u7 STP 794.0 GTC
...
```

```
2007/09/03  12:56:00  ATS/POSITION   ?BarCls        er2u7 792.5 792.5 792.5 792.5
2007/09/03  13:07:00  ATS/POSITION   ?AckFilled     BUY +100 er2u7 STP 792.8 (fa 793.0)
(Stop-loss filled asynchronously to a state of the ATS, position closed with the profit of $70)
2007/09/03  13:08:00  ATS/IDLE       ?BarCls        er2u7 793.0 793.0 793.0 793.0
2007/09/03  13:09:00  ATS/IDLE       ?BarCls        er2u7 793.7 794.0 793.7 794.0
...
2007-09-03  15:00:00  ATS/IDLE       ?BarCls        er2u7 793.2 793.2 793.2 793.2
2007-09-03  15:02:00  ATS/IDLE       ?BarCls        er2u7 793.2 793.2 793.2 793.2
(ATS decides to enter another short position)
2007-09-03  15:02:00  OE/IDLE        ?Order         SEL -100 er2u7
2007-09-03  15:03:00  ATS/PENDING_I  ?AckFilled     SEL -100 er2u7 (fa 793.5)
(Second position entered at 793.5, adding first stop-loss ...)
2007-09-03  15:03:00  OE/IDLE        ?Order         BUY +100 er2u7 STP 793.6
2007-09-03  15:04:00  ATS/POSITION   ?BarCls        er2u7 793.5 793.5 793.5 793.5
2007-09-03  15:09:00  ATS/POSITION   ?BarCls        er2u7 792.9 792.9 792.8 792.8
(Adding a new stop-loss in order to secure the accumulated profit ...)
2007-09-03  15:09:00  OE/IDLE        ?Cancel        BUY +100 er2u7 STP 793.6
2007-09-03  15:09:00  OE/IDLE        ?Order         BUY +100 er2u7 STP 793.2
2007-09-03  15:09:00  ATS/POSITION   ?AckCancelled  BUY +100 er2u7 STP 793.6
2007-09-03  15:14:00  ATS/POSITION   ?BarCls        er2u7 793.1 793.1 793.1 793.1
2007-09-03  15:16:00  ATS/POSITION   ?BarCls        er2u7 792.9 792.9 792.8 792.8
...
2007-09-03  15:21:00  ATS/POSITION   ?BarCls        er2u7 792.8 792.8 792.6 792.6
(Adding a new stop-loss in order to secure the accumulated profit ...)
2007-09-03  15:21:00  OE/IDLE        ?Cancel        BUY +100 er2u7 STP 793.2
2007-09-03  15:21:00  OE/IDLE        ?Order         BUY +100 er2u7 STP 793.1
2007-09-03  15:21:00  ATS/POSITION   ?AckCancelled  BUY +100 er2u7 STP 793.2
2007-09-03  15:23:00  ATS/POSITION   ?BarCls        er2u7 792.7 792.7 792.7 792.7
...
2007-09-03  15:26:00  ATS/POSITION   ?BarCls        er2u7 792.7 792.7 792.7 792.7
(Adding a new stop-loss in order to secure the accumulated profit ...)
2007-09-03  15:26:00  OE/IDLE        ?Cancel        BUY +100 er2u7 STP 793.1
2007-09-03  15:26:00  OE/IDLE        ?Order         BUY +100 er2u7 STP 793.0
2007-09-03  15:26:00  ATS/POSITION   ?AckCancelled  BUY +100 er2u7 STP 793.1
2007-09-03  15:30:00  ATS/POSITION   ?AckFilled     BUY +100 er2u7 STP 793.0 (fa 793.1)
(Stop-loss filled asynchronously to a state of the ATS, position closed with the profit of $40)
2007-09-03  15:31:00  ATS/IDLE       ?BarCls        er2u7 793.1 793.1 793.1 793.1
2007-09-03  15:32:00  ATS/IDLE       ?BarCls        er2u7 793.1 793.1 793.1 793.1
...
```

Since the raw output of the DEVS simulation doesn't provide very clear view to what happened during the simulation, such output is often transformed into a trade report and then statistically evaluated (as shown in sections 3.3 and 3.4).

Chapter 5

Optimization of trading strategies

Modeling and simulation of automated trading strategies allows traders to repeatedly and comparatively perform various strategies on the same data. As we have mentioned in the previous chapter, the simulation of trading strategies allows us to proceed also a wide spectrum of optimization tasks.

The optimization of trading strategies is a process of making them more profitable, robust and stable considering the given market conditions. This chapter introduces readers to the parametric trading strategies and shows how the genetic algorithms can be used for the optimization of them. This chapter contains also a case study of the genetic optimization and explains the biggest problem of all optimization methods which is the curve fitting of in-sample market data.

Since the purpose of this chapter is not to provide an exhaustive explanation of the theory of optimization, it provides only a limited knowledge related to this problematic from the application-specific point of view.

5.1 Parametric trading strategies

The initial concept of the optimization is based on a core of the trading strategy. A programmer manually defines such core and then he lets the optimizer to optimize parameters in order to find the best parametric strategy.

The parametric strategy is a class that implements an interface extended from the interface of a standard strategy (see Code 5.1). Each parametric strategy contains a mandatory array of parameters P, which affects both its behavior and overall outcome.

Code 5.1:

```
interface ParametricStrategyInterface extends StrategyInterface
begin
   methods:
      setParameters( P )
      getParameters()
      initParameters()
end
```

The array P contains double and integer parameters with associated allowed intervals.

The integer parameter $p_i \in P$ is a triple $p_i = (v, v_{\min}, v_{\max})$, where $v \in \mathbb{Z} \cap \langle v_{\min}, v_{\max} \rangle$. Similarly, the double parameter $p_d \in P$ is defined as $p_v = (v, v_{\min}, v_{\max})$, where

$v \in \mathbb{R} \cap \langle v_{\min}, v_{\max} \rangle$. The intervals of allowed values $\langle v_{\min}, v_{\max} \rangle$ determine a parameter space, in which the optimization algorithm searches the solution.

Let's consider a simple trend-following parametric strategy which takes entry decisions according to the crossover of two different simple moving averages (SSMA, LSMA). The strategy exits are defined just by the money management rules (profit targets and stop-losses). A core of the strategy looks as shown in Code 5.2.

Code 5.2:

```
class MyParametricStrategy extends ParametricStrategy
                           implements ParametricStrategyInterface
begin
   variables:
      P[SSMA,LSMA,PT,SL]:array of Parameter        // An array of parameters
   methods:
      method initParameters()
      begin
         let P[SSMA] = (null,3,15),  P[LSMA] = (null,12,32)
         let P[PT] = (null,0.25,20.00),  P[SL] = (null,0.25,5.00)
      end
      method outOfPositionBarClosed(t)
      begin
         if SMA_P[SSMA] crosses SMA_P[LSMA] from the bottom then
            enter long position
         if SMA_P[SSMA] crosses SMA_P[LSMA] from the top then
            enter short positon
      end
      method positionEntered(t)
      begin
         enter stop-loss order P[SL] above/below the entry price
         enter profit target order P[PT] above/below the entry price
      end
end
```

The optimizer gets such strategy and automatically optimizes the array of parameters P in order to maximize its fitness value. Because of this, the quality of a whole optimization process depends on a used fitness function. We will optimize the same strategy on the same data using four different fitness functions proposed in the equations 5.1.a-d to illustrate the importance of their proper selection.

Considering the given market data, the solution of an optimization problem is determined by a core of the trading strategy and by the array of optimal parameters P_{opt}. The fitness of such solution is computed from the strategy statistics, which has been proposed in Section 3.3.

5.1.1 Choosing an appropriate fitness function

We have explained in the third chapter that the absolute outcome is not always the most significant indicator used in the evaluation of trading strategies. Another important indicator is

strategy robustness, because it allows us to employ bigger financial leverage compensating smaller absolute profits. Equations 5.1.a-d illustrate several fitness functions and indicators.

$$\text{fitness}(P) = efi \tag{5.1.a}$$

$$\text{fitness}(P) = ehi - maxad \tag{5.1.b}$$

$$\text{fitness}(P) = |awi| - |alo| \tag{5.1.c}$$

$$\text{fitness}(P) = \frac{nwi}{ntr}|awi| - \frac{nlo}{ntr}|llo| \tag{5.1.d}$$

The *efi* is an abbreviation of the *final equity*, *awi* and *alo* stands for the *average win* and *average loss*, *llo* for the *largest loss*, *nwi*, *nlo* and *ntr* for the *numbers of wins, losses* and *trades* and *maxad* for the *maximum account drawdown*.

Function 5.1.a evaluates only the absolute outcome of a strategy (the final equity). The main disadvantage of this function is that it doesn't regard robustness and stability of trading strategies.

The robustness can be evaluated in many different ways. The first way is as follows. The sequence of trades $t_0, t_1, \ldots, t_{n-1}$ from the trade report is divided into several subsequences (for example into $t_0 \ldots t_9; t_{10} \ldots t_{19}; t_{20} \ldots t_{29}; t_{30} \ldots t_{n-1}$). Then, particular outcomes of the subsequences are computed as $\text{account}(t_{i+10}) - \text{account}(t_i)$ for $i = 0, 10, 20, \ldots$, where $\text{account}(t_i)$ is the amount of financial resources available on the trading account after the i^{th} trade. Finally, the standard deviation of these particular outcomes is calculated and its inverse is then regarded as the indicator of strategy robustness (Eq. 5.2).

$$\text{robustness}\left(\left[t_i\right]_{i=0}^{n-1}\right) = \text{std}\left[\text{account}(t_{i+10}) - \text{account}(t_i)\right]_{i=0,10,\ldots}^{n-1} \tag{5.2}$$

We will use the second, much simpler approach for the computation of strategy robustness. It is given simply as a difference between the highest equity and the maximum drawdown of the account. This principle is involved also in the fitness function shown in Equation 5.1.b.

The third fitness function regards only the average win and average loss (Eq. 5.1.c). These variables correspond to a concept of the risk-reward ratio. Since this function doesn't consider the accuracy of a strategy's signaling system (win-loss ratio), it produces very bad results. This problem is solved by the fourth fitness function (Eq. 5.1.d), which considers both risk-reward ratio and signaling system accuracy.

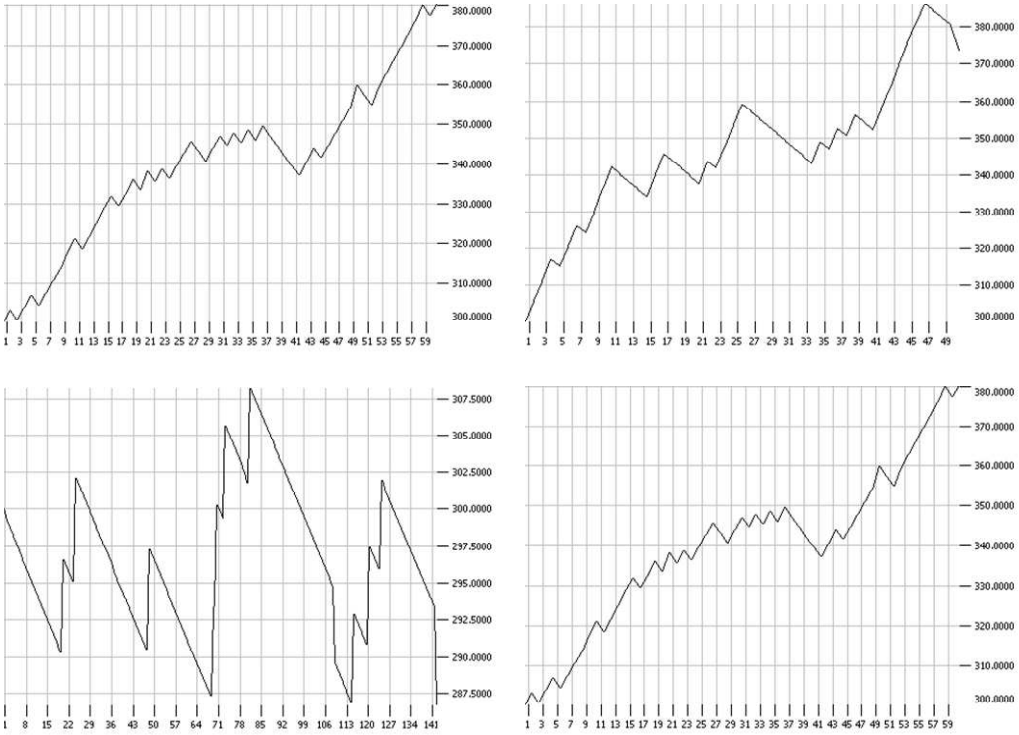
We have optimized our strategy for the intraday data of the E-mini Russell 2000 futures using four different fitness functions. The simulation starts at the market opening on the 1st of September 2007 and continues until the 15th of the same month.

The following table contains the results of such optimizations.

Fitness function	LSMA	SSMA	PT	SL
$\text{fitness}(P) = efi$	16	8	3	2.5
$\text{fitness}(P) = ehi - maxad$	16	8	5.5	0.5
$\text{fitness}(P) = \lvert awi \rvert - \lvert alo \rvert$	16	12	6	0.5

$\text{fitness}(P) = \frac{nwi}{ntr}\lvert awi\rvert - \frac{nlo}{ntr}\lvert llo\rvert$	16	8	3	2.5

Figures 5.1.a, 5.1.b, 5.1.c and 5.1.d show equity curves of the corresponding strategies. Starting with the capital of $300, the strategies optimized using the first and fourth fitness function generated a profit of $80. The second fitness function, which involves the simplified concept of strategy robustness, generated a profit of $74. Although this profit is smaller, the strategy optimized by the second fitness function uses a money management that holds positions longer. Considering various fees and commissions, the smaller number of total trades can be regarded as an advantage. As we have expected, the worst results gives the third fitness function. We can see in Figure 5.1.c that the resulting strategy is very unstable because it drew down the accumulated profit several times during the simulation.



A	B
C	D

Figure 5.1: Various equity curves of the optimized trading strategy. Each curve corresponds to a different fitness function used during the optimization.

5.1.2 Parametric surface

The parametric surface is the *n*-dimensional function that evaluates the fitness of a set of n parameters. The optimization algorithm iteratively tries to find a maximum of such surface (or a minimum if the fitness function is replaced by the cost function). In our case, the surface is determined by results of the trading simulation.

Now, we take a closer look to the second fitness function (see Eq. 5.1.b). This function considers both absolute outcome and stability of a trading strategy. The simulator iteratively evaluates the fitness function for all combinations of parameters. If we assume that lengths of

moving averages in our strategy are examined with the step of one and money management price levels with the step of 0.25, the resulting number of configurations which must be evaluated is given as $13 \times 21 \times 80 \times 20 = 436800$. Although the four-dimensional parameter space can be fully examined using standard computational resources, the bigger spaces must be processed by more sophisticated optimization methods, such as by genetic algorithms or others.

The systematic examination of the parameter space produces the following table:

LSMA	SSMA	PT	SL	fitness
14	12	1.5	0.5	315,315
16	12	1.5	0.5	286,596
14	8	2.0	0.5	326,547
16	8	2.0	0.5	329,263
14	10	2.5	0.5	296,548
16	10	2.5	0.5	288,993

...

If we transfer these data into a pivot chart, we can evaluate how the individual parameters affect the fitness of a whole strategy. Since the four-dimensional surface cannot be illustrated graphically, we transfer it into two different pivot charts. The first chart covers parameters of the signaling system, which decides whether to enter the position (see Fig. 5.2).

These parameters include the longer (LSMA) and shorter (SSMA) simple moving average. As we can see, the length of the shorter SMA doesn't affect the overall performance of the trading strategy, but the strategy works well only if the length of the longer SMA is less than 18 bars. Of course, this fact doesn't mean that the er2u7 market doesn't contain trends with a bigger period. This just means that entries based on longer moving averages are not compatible with exit methods based on tight money management stops.

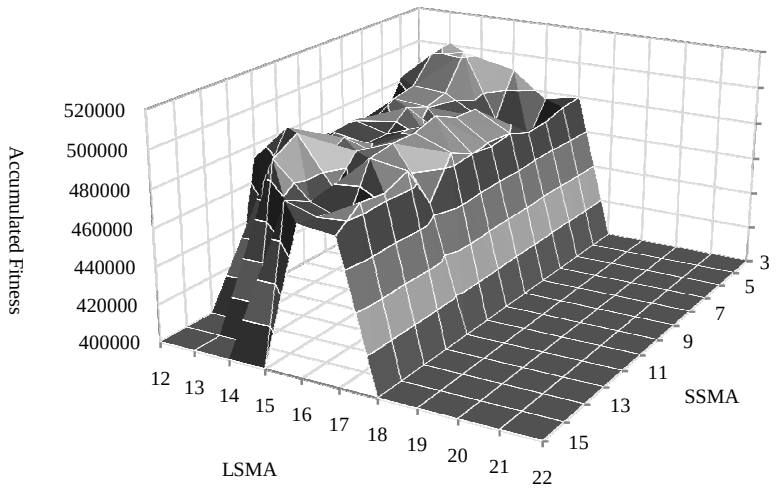


Figure 5.2: The pivot chart that illustrates dependencies between parameters of the signaling system and the overall fitness of our strategy.

The second pivot chart covers money management parameters, such as the profit target (PT) and the stop-loss (SL). These parameters affect mainly the conditions at which the money management exits the position. We have mentioned in Section 3.2 that there exist several

dependencies between the Risk-Reward Ratio determined by sizes of the profit targets and stop-losses and the probability that these targets will be hit.

Considering Figure 5.3, our strategy gives the best outcome in a combination with the moderate profit targets and stop-losses (profit target of $5.75 and stop loss of $5.0). The worst outcome gives the combination of parameters that leads to a very big Risk-Reward Ratio (for example PT of $5.75 and SL of $0.5). This is caused by a fact that most of the trades are exited at tight stop-losses and big profit targets are never hit.

Please note that the real trading involves various fees and commissions which are comparable to sizes of our profit targets. Because of this, such tight money management stops are unsuitable for the real trading.

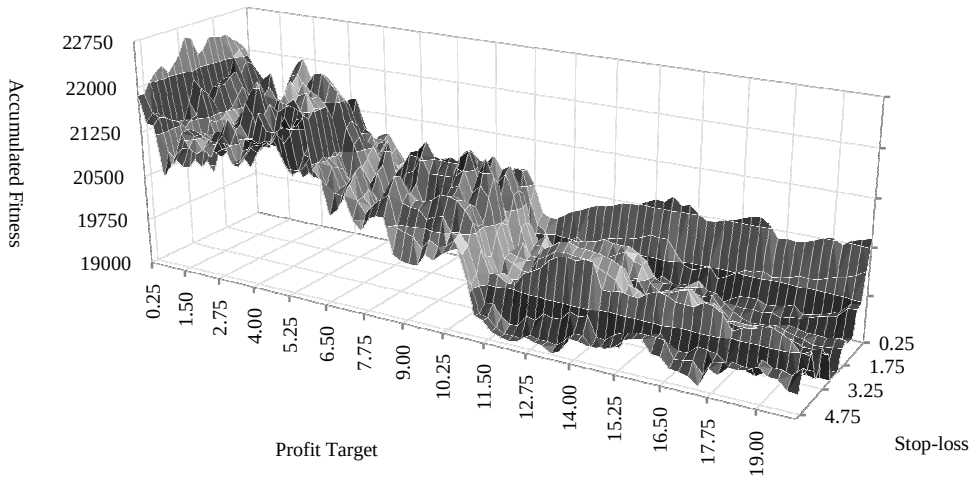


Figure 5.3: The pivot chart that illustrates dependencies between the size of money management stops (PT, SL) and the overall fitness of our strategy.

5.2 Exhaustive search

The exhaustive search is a brute force method that optimizes strategy by evaluating all combinations of its parameters. This solution is suitable for both continuous and discrete parameter spaces and doesn't get stuck in local maximums. The main problem of the exhaustive search is that the exploration of the whole parameter space is a very time consuming task, appropriate only for low-dimensional topologies.

The improved exhaustive search involves an adaptive sampling of the parameter space. This approach assumes that the fitness function is continuous with a limited derivative for every combination of parameters. In other words, the corresponding fitness surface has a limited slope in all its points. The adaptive sampling works as follows. Firstly, the whole fitness surface is roughly mapped by the brute force algorithm using a lower resolution. After this, the algorithm selects areas with higher values of the fitness function and examines them more precisely at the higher resolution. The pyramidal implementation of a brute force approach allows us to examine wide multidimensional surfaces in the acceptable time.

R. Haupt illustrates this principle in his book (17) using the concept of a helicopter view. If we want to find the place with a lowest altitude in the Grand Canyon, we use a helicopter to

take the brief overview of a terrain. After that, we survey the valleys, but not the ridges and peaks. The speed of such exploration dramatically rises, similarly as we save much time by the pyramidal implementation of an exhaustive search optimizer.

5.3 Genetic algorithms

The previous optimization methods have been intended to a systematic exhaustive exploration of the parameter space. Similar approaches are sometimes referred as *blind*, because they examine the whole fitness surface in all directions from the starting point without respecting a slope of the surface.

Other methods, which strictly follow the gradient of a surface sometimes stuck in local maximums. The second disadvantage is that they require an analytical derivative of the fitness surface, which isn't always available. For example, the fitness surface of a strategy parameter space is determined by the simulation process, thus it cannot be described by an analytical function.

We can say that genetic algorithms lack disadvantages of both exhaustive search and analytical optimization methods. As we will see later, a genetic algorithm is not a blind method, because only the individuals with a higher fitness value survive. This principle significantly reduces the number of configurations which must be examined.

Secondly, the genetic algorithms usually don't tend to stuck in local maximums and don't require an analytical description of the fitness function's derivative.

The purpose of this section is not to provide an exhaustive description of all aspects of the genetic optimization. Because of this, it is not concerned to various competitive parent selection approaches, recombination schemes and encoding principles.

5.3.1 Inspiration from nature

Genetic algorithms as well as the evolutionary optimization techniques have been inspired by natural processes and living organisms. In nature, every organism is characterized by the sequence of genes, which is carried in chromosomes in a form of the DNA (Deoxyribonucleic acid). The DNA molecules store all information needed to construct other cell components and to determine all traits of an individual, which are given just by the combination of genes contained in the DNA sequence.

The Charles Darwin's theory of evolution assumes several basic premises. At first, offspring inherits many of the traits of its parents and the population as a whole is stable. Secondly, only a small part of offspring survives to the full age. The probability that a certain individual survives to the adulthood is given by the compliance of its traits with the environment in which such individual lives.

The process of evolution is based on an iterative replacement of the parent population by its offspring. The replacement is performed by the operations of recombination and mutation. The recombination means that the gene sequence of a child is derived from the sequences of both parents by the exchange of genetic information. The mutation is a change of the genetic material in a certain organism caused for example by an ionizing or ultraviolet radiation.

The principles of the recombination, mutation and genetic evolution are pylons of the evolutionary theory. As we will see in the further text, these principles have been successfully adopted also in the computer science and optimization algorithms.

5.3.2 Computational model of the genetic evolution

The computational model of the genetic evolution simulates processes from the nature described in the Darwin's evolutionary theory. The computational model involves two aspects which are unique for every optimization task.

The first aspect is a fitness function. Since we have introduced various fitness functions in Section 5.1.1, we think that there isn't a need to discuss them in this section.

The second aspect is a genetic representation of the solution domain. The solution domain as well as its genetic representation is a problem that depends on a concrete optimization task. For example, the permutation tasks require a different genetic representation than the equation root-finding problems.

Considering the optimization of trading strategies, the solution domain is given by the parameter space of a concrete strategy. Because of this, genes of a chromosome correspond to the strategy parameters. Trading strategies allow both continuous and discrete parameters, where continuous parameters are represented by floating-point numbers and discrete ones by integer values.

Genetic representation of the trading strategy

Let's assume the same strategy as we have proposed in Code 5.2. The signaling system will consist of two moving averages which determine whether to enter the position and of mandatory stop-losses and profit targets. Since genetic algorithms allow us to use much wider parametric spaces, we modify the range of strategy parameters as shown in the following table:

Parameter	Min-value	Max-value
LSMA	12	32
SSMA	3	15
PT	0.25	20
SL	0.25	5

A chromosome of such strategy will be then encoded as a sequence of four values (Eq. 5.3). The moving averages are encoded as integer values, whereas the money management stops are represented by floating point numbers.

$$(\mathrm{LSMA},\mathrm{SSMA},\mathrm{PT},\mathrm{SL}) \tag{5.3}$$

The flow chart

Now, when we have introduced the genetic representation of a trading strategy, we can explain how the genetic information is altered in order to evolve the best trading strategy. The iterative process employs the selection and combination of parents, creation of a new generation and also the mutation of selected individuals. This process can be illustrated by a flow chart, as shown in Figure 5.4.

At first, the optimization process creates a new initial population with a predefined size. Chromosomes in the population are generated randomly, respecting the constraints of individual genes (parameters). Code 5.2 shows how these constraints are defined in a trading strategy.

After the new population is created, the process evaluates fitness of all chromosomes in the population. This is done by systematic repeated simulations of a trading strategy. Because of

this, the simulation must be performed for every single chromosome. This is the reason why the evaluation of fitness is the most time-consuming step in a whole optimization task.

The third step is a selection of mates from the parent population. Although there are many different competitive approaches of the selection of mates, we don't think that the exhaustive comparison of them follows the topic of this thesis. All selection methods are based on an assumption that more reliable and stronger individuals have a bigger chance to survive and to reproduce.

The fourth step is a combination of mates. This means that the individuals selected in the previous step are mated together in order to produce an equal number of children. Since the total number of individuals in a population must remain unchanged, the newly generated children replace a worst part of the old population.

The last step is a mutation of children. The mutation brings new properties and traits into the population.

The evolution process iteratively continues until the end condition is satisfied. The optimization ends when the population converges, or after a predefined number of iterations.

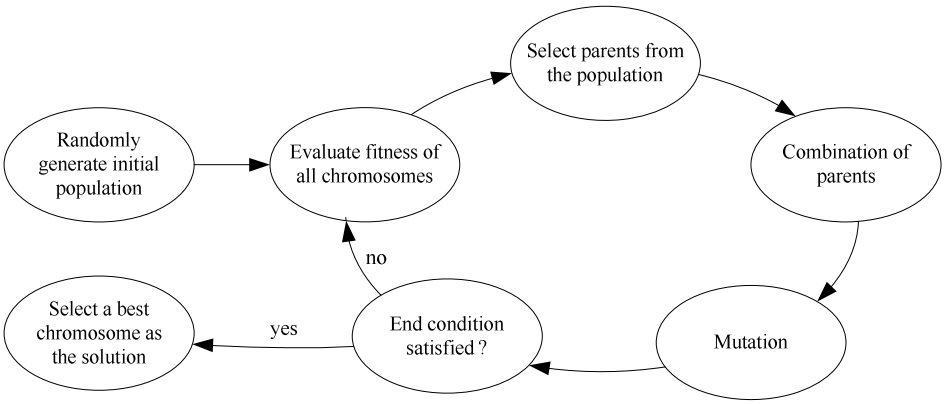


Figure 5.4: The flow chart of the process of genetic evolution.

Selection of parents

As we have said, the common assumption for all methods of the parent selection is that more reliable and stronger individual in nature has a bigger chance to reproduce. Although there are many advanced approaches which select individuals with a probability of ranked fitness functions, we think that a simple *Elite Selection* method will be sufficient for our purposes.

This method simply sorts individuals in a population by its fitness value. After that, it selects a certain number of best individuals for a reproduction (Code 5.3). Let's assume that we have a population of the size of n. Generally, we select m best individuals for a reproduction, where $1 < m \leq n$. We can theoretically select only one pair as well as the whole population, but we usually mate best 5% - 40% of all individuals.

Code 5.3:

```
method selectParents( m , [P_i]_{i=0}^{n-1} )
begin
   descendingly sort [P_i]_{i=0}^{n-1} by fitness(P_i)
```

```
    return [P_i]_{i=0}^{m-1}
end
```

Recombination

The *Elite Selection* method selects parents for the recombination, but it doesn't designate individual mates which will be mutually recombined. Since the mutual reciprocation of the parent chromosomes doesn't matter, the recombination algorithm selects mates randomly.

When the series of chromosomes $\left[P_i\right]_{i=0}^{n-1}$ is descendingly sorted according to the fitness function, the recombination algorithm randomly fetches two mates from the subsequence $\left[P_i\right]_{i=0}^{m-1}$, where $m \leq n$. After that, it recombines and moves them to the end of the sequence $\left[P_i\right]_{i=0}^{n-1}$. Finally, the algorithm decreases the variable m and the whole process continues until all parents are recombined (Fig. 5.5).

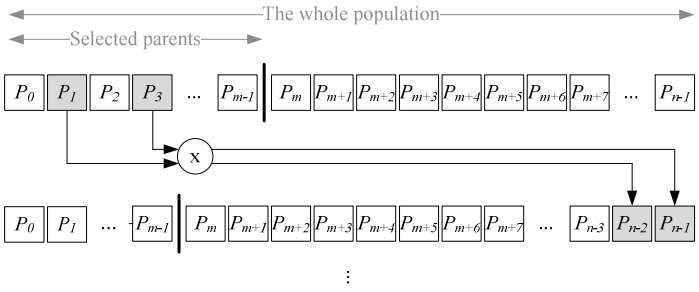


Figure 5.5: The iterative replacement of the part of parent population by offspring.

The recombination of two mates is based on a double crossover. Considering a gene sequence with the length of n, the recombination algorithm randomly selects two crossover points x_1 and x_2 from the interval $\langle 0, n \rangle$, where $x_1 < x_2$.

If we denote the first chromosome as $P^A = \left(p_0^A, p_1^A, \ldots, p_{n-1}^A\right)$ and the second as $P^B = \left(p_0^B, p_1^B, \ldots, p_{n-1}^B\right)$, the recombination in points x_1 and x_2 will produce the following offspring (Eq. 5.4):

$$P^C = \left(p_0^C, p_1^C, \ldots, p_{n-1}^C\right), P^D = \left(p_0^D, p_1^D, \ldots, p_{n-1}^D\right), \tag{5.4}$$

where:

$$p_i^C = \begin{cases} p_i^A & \text{if} \quad x_1 \leq i < x_2 \\ p_i^B & \text{in other cases} \end{cases} \text{ and } p_i^D = \begin{cases} p_i^B & \text{if} \quad x_1 \leq i < x_2 \\ p_i^A & \text{in other cases} \end{cases} \tag{5.5}$$

The principle of a double crossover can be graphically illustrated as shown in Figure 5.6:

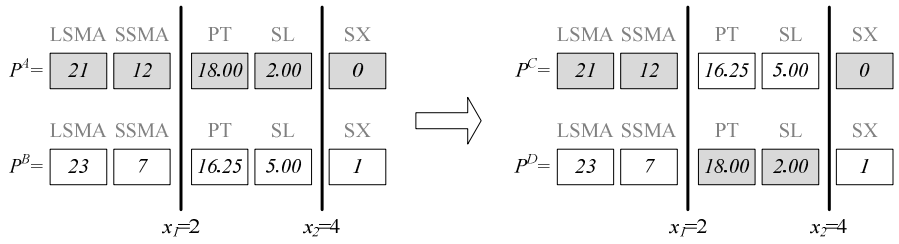


Figure 5.6: The double crossover of two parent chromosomes.

Mutation

The mutation of offspring is a very important step that brings new traits into the population. Without this, the resulting chromosome would be still only a combination of the initial population, nevertheless how much iterations the optimization algorithm has performed.

Because of this, the algorithm randomly selects several individuals from the offspring population and randomly mutates several genes within their chromosomes.

The process of mutation has three different parameters. The first is a *Chromosome Percentage* c which indicates how many chromosomes will be mutated. The second parameter, *Gene Percentage* g defines how many genes within a selected chromosome will be affected by the mutation. The third parameter v indicates a maximum percentage change of the gene value. For example, if the old value of a gene is $p_i^{old} = 0.25$ and $v = 0.3$, then the new gene value after the mutation will be somewhere in following interval (Eq. 5.6).

$$\begin{aligned} p_i^{new} &= \left\langle p_i^{old} \times (1-v),\, p_i^{old} \times (1+v) \right\rangle = \\ &= \left\langle 0.25 \times (1-0.3),\, 0.25 \times (1+0.3) \right\rangle = \\ &= \left\langle 0.175, 0.325 \right\rangle \end{aligned} \tag{5.6}$$

5.3.3 Optimization case study

In this case study, we run the optimization algorithm on the parametric strategy proposed in Section 5.1. The parametric surface that corresponds to this strategy is very wide and covers all 436800 configurations. It has taken several days to fully examine the whole parametric space and to render the parametric surface as shown in figures 5.2 and 5.3.

Now, we will try to solve this problem by a genetic algorithm. The algorithm generates an initial population with the size of 65 chromosomes and performs 60 iterations respecting the computational model of the genetic evolution. The process selects best 10% of the population for a reproduction. The mutation rates are set as follows: $c = 0.15$, $g = 0.6$ and $v = 0.3$.

Figure 5.7 illustrates how the optimization process converges to the optimal solution. We can see that a stable solution is found just after 35 iterations of the algorithm. The fitness function of such solution is 442.48.

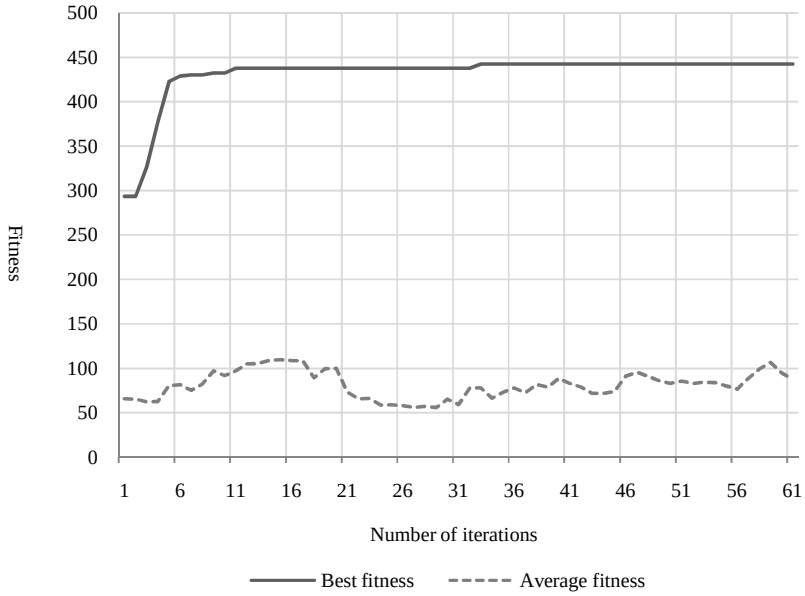


Figure 5.7: The convergence of the trading strategy genetic optimization.

The biggest problem of all optimization techniques is that they optimize trading strategies regarding a given sample of market data. These data are often referred as *training* or *in-sampl*e. An optimization algorithm finds the optimal solution, but it doesn't evaluate how stable this solution will be if market conditions change.

Because of this, the successfulness of trading strategies is often tested on the *out-of-sample* (*testing*) data, which may potentially contain different market conditions. The optimized trading strategy can be considered as robust, if its out-of-sample performance has the same characteristics as the performance based on the training data.

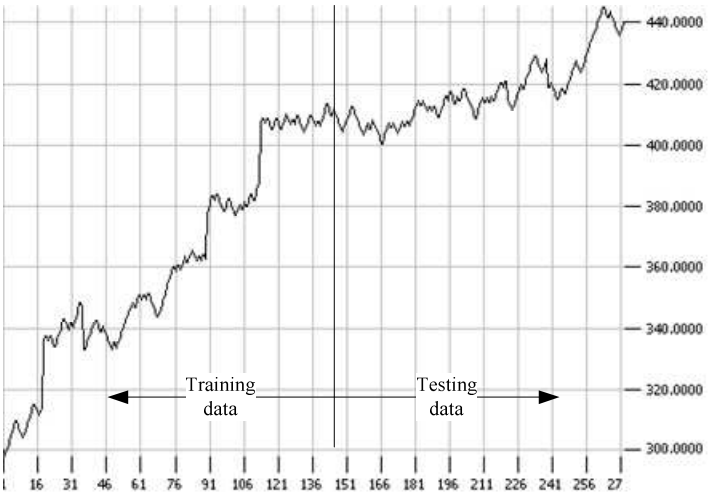


Figure 5.8: The performance of the trading strategy simulated on training and testing (out-of-sample) data.

Figure 5.8 contains the equity curve of our strategy. The strategy has been optimized and tested on the E-mini Russell 2000 futures market data with the expiration month of September 2007.

The training has been performed on the data between the 15th of August 2007 and 2nd of September 2007. After that, the strategy has been tested on the out-of-sample data until the expiration date of 21th of September 2007.

As we can see in Figure 5.8, our strategy wasn't able to keep its previous performance on the testing data. The strategy generated 142 in-sample and 130 out-of-sample trades with total returns of 27.32% and 10.45%, respectively.

The following table contains a statistical comparison of both simulations:

Simulation on training data		Simulation on testing data	
Start	2007-08-15 16:53:04	Start	2007-09-03 16:22:04
End	2007-08-31 21:49:29	End	2007-09-21 09:22:00
Number of trades	142	Number of trades	130
Number of wins	83	Number of wins	71
Number of losses	59	Number of losses	59
Number of profit target exits	83	Number of profit target exits	71
Number of stop-loss exits	59	Number of stop-loss exits	59
Number of strategy exits	0	Number of strategy exits	0
Average number of stop-loss moves	0.00	Average number of stop-loss moves	0.00
Max contracts per trade	1	Max contracts per trade	1
Min contracts per trade	1	Min contracts per trade	1
Avg contracts per trade	1.00	Avg contracts per trade	1.00
Equity start	412.76	Equity start	300.00
Equity high	525.52	Equity high	336.33
Equity low	410.76	Equity low	291.23
Final equity	525.52	Final equity	331.34
Return of starting equity	27.32%	Return of starting equity	10.45%
Largest win	22.30	Largest win	3.39
Largest loss	-14.16	Largest loss	-9.63
Average win	2.93	Average win	2.26
Average loss	-2.21	Average loss	-2.19
Max consecutive wins	7	Max consecutive wins	10
Max consecutive losses	4	Max consecutive losses	5
Average profit excursion	2.07	Average profit excursion	1.56
Average adverse excursion	-1.35	Average adverse excursion	-1.36
Max profit excursion	22.30	Max profit excursion	3.39
Max adverse excursion	-14.16	Max adverse excursion	-9.63
Max drawdown	16.16	Max drawdown	14.71
Max position duration	2 dys 39 hrs	Max position duration	13 hrs 52 min
Min position duration	16 sec 28 ms	Min position duration	24 sec 97 ms
Avg position duration	1 hrs 36 min	Avg position duration	1 hrs 27 min
Annual Return	22936.41%	Annual Return	675.24%

If we compare individual characteristics, such as average wins, average losses, win/loss ratio, we find out that they don't differ much in both the cases. The effect of a compound interest causes that small differences in individual trades cause a big difference in results of the whole simulation.

According to the statistics, the extrapolated annual return of the strategy simulated on the training data is a little over twenty thousand percents. It is clear that if we had the strategy which would be able to continually gain such annual returns, we would be soon the world's wealthiest person.[27] This is of course a very absurd vision just for two reasons.

At first, financial markets are very dynamic places, where it is impossible to continually gain the same profits with unmodified trading methods over a long period of time. The optimization of trading strategies on the in-sample data is, in fact, the extrapolation of past into the future.

Secondly, the capitalization of revenues and the compound interest causes that every trading operation will later carry a very huge amount of financial resources. Although the world financial markets are very liquid places, the immediate transaction of several stocks doesn't necessarily mean that the transaction of several million dollars will be also as immediate.

[27] Considering the starting capital of $1000, it would take less than four years.

Chapter 6

Fuzzy approach to trading strategies

Automated trading systems are based mainly on a computational evaluation of crisp inputs. Even though this evaluation is the objective process that eliminates psychological aspects of the trading, it is not very robust since it doesn't reflect well the vagueness of the real trading environment. This is caused by a fact that conventional automated trading strategies are based on an exact and symbolic representation of the reality. As we have said in the introduction of this thesis, such applications bring some difficulties to their interaction with the vague and indeterminate environment. Since this, we have sketched a soft computing approach as the computational model for a wide range of economic processes.

In this chapter, we will show how the soft computing methods, especially fuzzy logic, can be used for a more reliable prediction and modeling of market behavior. Lotfi Zadeh states that the guiding principle of the soft computing is "*to exploit the tolerance for imprecision, uncertainty and partial truth to achieve tractability, robustness, low solution cost and better rapport with reality. In the final analysis, the role model for soft computing is the human mind.*" (18)

We have also explained in the third chapter why hard computing trading strategies lack robustness due to the sensitivity to changes of parameters. This chapter explains basic principles of fuzzy logic and fuzzy reasoning and shows how these principles can be applied in a construction of robust trading strategies.

Although the understanding of the fuzzy concept is a key prerequisite for this chapter, we won't go into particulars. Considering a big amount of specialized literature, we don't think that the exhaustive explanation of this concept is necessary for this chapter.[28]

6.1 Concept of uncertainty and the basics of fuzzy logic theory

Several decades ago, scientists and researchers believed that every problem, no matter how complex it is, can be solved by a computational system with an adequate power. These systems employ analytical methods which are based upon calculus, and thus require every single variable of the problem to be known. Because of this, such approach faces two difficulties.

The first is that the computational power required for an analytical solution of most real problems is beyond our resources. The exact analytical description of real problems may involve millions of different circumstances, which we usually don't know. This is the second

[28] For further reading, we recommend the Klir and Yuan's book (19) that comprehensively explains fuzzy sets and fuzzy logic from the both theoretical and practical points of view.

difficulty that makes the analytical solution of real problems practically impossible. This lack of information is referred as the uncertainty of the environment.

The uncertainty found everywhere in nature forced scientists to reevaluate their approaches and attitudes to solving large, complex problems. The concept of uncertainty originates from the shift of paradigm in artificial intelligence. If we are not able to avoid the uncertainty, we must use computational methods which implicitly allow the uncertainty in input data. This paradigm doesn't consider uncertainty as a disadvantage, because it reduces the complexity of problems which are unsolvable by traditional hard computing methods.

The group of methodologies which accept uncertainty in data is covered by the term *soft computing*. One of these methodologies is a theory of fuzzy sets.

The theory of fuzzy sets is inspired by a way how humans deal with the uncertainty. Humans often employ uncertainty in both their reasoning and communication. Despite they usually don't know all circumstances of the solved problem; they are able to properly evaluate such situation and to take the right decisions.

This principle can be illustrated by the following example. Let's imagine that we are cooking by a book. Many cooking books contain only a vague sequence of steps needed to cook the meal. Although we don't exactly know how much the "cup of flour" or "a little salt" means, we are able to prepare a good meal using such vague recipe. The fuzziness of information is sometimes eligible, because it significantly simplifies the process of reasoning.

The computational model which is able to work with such data is based on fuzzy logic and fuzzy sets. As we will see later, a fuzzy computational model performs operations on linguistic values (as humans do), rather than on crisp numerical values.

6.1.1 Linguistic variables and fuzzy sets

The fuzzy set is a concept which allows us to employ uncertainty in our computational models. The classical approach that doesn't allow uncertainty is based on crisp sets, which are determined strictly by the corresponding elements. For each element of the universe, it is clear when the crisp set contains such element and when it doesn't. For example, the set of all employees of the XYZ inc. is crisp, because there are only two strictly distinguishable possibilities – a man is or is not an employee of such firm.

Considering another real example, the set of all experienced employees of a certain firm is not crisp, because the definition of an experienced employee may vary. Is an employee experienced if it has at least two, five or even ten years of experience? Although the correct answer depends on a definition, we can say that the employee with five years of experience is more experienced that another one, which has only two years.

If we assume an example from the trading environment, we will find out that sets of all buying and selling opportunities are also not crisp. We can dispute which of two market opportunities is more appropriate for buying, but we cannot strictly distinguish between the advisable and unadvisable market opportunities.[29]

Because of this, we use a degree of membership as the measurement of how well an element fits the definition and purpose of a set. Such defined set is then referred as the fuzzy set.

[29] Although the fuzzy reasoning process doesn't strictly distinguish between the advisable and unadvisable market opportunities, the automatic trading system which implements such process must finally decide whether to or not to enter the position. As we will see later, this decision is taken in the last step of the reasoning process called *defuzzification*.

Crisp sets have only two degrees of membership – "1" for elements contained in the set and "0" for others. It would be very obscure if an employee with the experience of exactly three years was regarded as *experienced*, while the experience of two years and eleven months fell into a completely different set. Because of this, degrees of membership as well as degrees of truth in fuzzy sets are expressed by values from the continuous interval $\langle 0,1 \rangle$.

For example, let's establish a linguistic variable *market assessment* with three different linguistic terms – *appropriate for selling*, *appropriate for staying away* and *appropriate for buying*. Each linguistic term corresponds to a fuzzy set that contains elements described by such term. The degree of membership tells us how well a linguistic term fits properties of the underlying element (Fig. 6.1).[30]

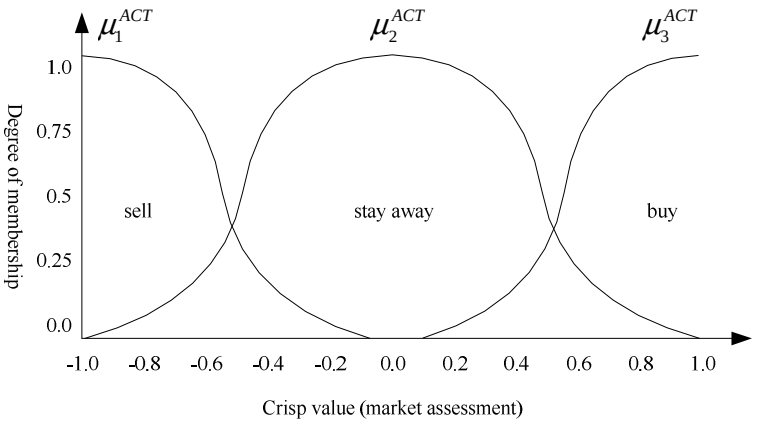


Figure 6.1: Membership functions of the linguistic variable *market assessment*.

The theory of crisp sets defines standard operations of intersection, union and complement (equations 6.1.a-c). Considering that A and B are crisp sets, these operations are defined as follows:

$$A \cap B = \{x | x \in A \wedge x \in B\} \tag{6.1.a}$$

$$A \cup B = \{x | x \in A \vee x \in B\} \tag{6.1.b}$$

$$\neg A = \{x | x \notin A\} \tag{6.1.c}$$

The analogical operations can be defined also for fuzzy sets. The fuzzy set is determined by a membership function that assigns the degree of membership to each element in the universe. If μ_A and μ_B are membership functions that determine fuzzy sets A and B, the operations of intersection, union and complement will be then defined as shown in equations 6.2.a-c.[31]

[30] The selection of linguistic variables, terms and membership functions is an application-specific problem. Although we have illustrated membership functions with the shape of Gaussian bell, there exist also many other possible shapes, such as triangles, trapezoids, L-functions and others.

[31] Operations on fuzzy sets can be defined in many different ways. In general, these operations are referred as *s-norms* (union) and *t-norms* (intersection). The min/max norm proposed in equations 7.2.a and 6.2.b is only one of these ways.

$$\mu_{A\cap B}(x) = \min\left(\mu_A(x), \mu_B(x)\right) \tag{6.2.a}$$

$$\mu_{A\cup B}(x) = \max\left(\mu_A(x), \mu_B(x)\right) \tag{6.2.b}$$

$$\mu_{\neg A}(x) = 1 - \mu_A(x) \tag{6.2.c}$$

Let A and B to be crisp sets defined on two different universes. The Cartesian product of such sets is defined as shown in Equation 6.3:

$$A \times B = \left\{(x, y) \middle| x \in A \wedge y \in B\right\} \tag{7.3}$$

The Cartesian product of two fuzzy sets is also a fuzzy set, specifically the set of duples (x, y), where the corresponding membership function is defined by the following equation:

$$\mu_{A\times B}(x, y) = \min\left(\mu_A(x), \mu_B(y)\right) \tag{7.4}$$

Since logical operations on fuzzy propositions directly correspond to the operations on fuzzy sets, we will build on these principles in the next section.

Linguistic variables and linguistic terms are building blocks for reasoning rules which involve uncertainty. Because fuzzy rules cannot be processed by classical hard computing algorithms, we must involve so-called *fuzzy inference systems* to implement logical reasoning with the uncertainty.

6.2 Fuzzy-based trading strategies

We have illustrated in Section 3.4 how parameters of a trading strategy affect its robustness. The described sensitivity is generally unwanted and indicates that a strategy lacks robustness. Although various optimization methods can boost the performance on an in-sample data, the out-of-sample profitability is not assured. A strategy can assure the out-of-sample profitability only in the case if it's in-sample outcome doesn't depend on the parameters.

Since almost all of the trading strategies are parameterized, traders must usually adjust parameters of a generic strategy to fit it to concrete market conditions. In this section, we will illustrate how the fuzzy approach can be used in the Triple Screen Trading System to improve its robustness.

6.2.1 Triple Screen Trading System

The conventional Triple Screen Trading System was developed by Alexander Elder in 1985 (14). It is a compound system that combines both trend and countertrend indicators together to provide more reliable signals.

The Triple Screen Trading System is based on two major paradigms. The first is an assumption that markets have a trending behavior. This assumption is supported by the well-known slogan "*Trend is your friend*". Trading methods based on this paradigm regard the market direction rather than the market state.

The second paradigm assumes that short-term market swings follow the long-term movements. Considering the long-term market state, this type of swings causes that markets can

be temporarily overbold or oversold. Because of this, the counter-trend style of trading regards the market state rather than its direction. The supporting slogan for this style is "*Buy low, sell high*".

The first paradigm is covered by trend-following indicators, such as moving averages or convergence/divergence oscillators, while the second one corresponds to counter-trend oscillators, such as RSI, Williams %R, Bollinger Bands and others.

The only problem is that trend-following and counter-trend tools generate opposite signals on the same timeframe. Because of this, the Triple Screen Trading System uses several timeframes (or screens) for the analysis of market situation. This system in its original version involves weekly timeframe on the first screen and daily timeframe on the second screen. However, the generalized version employs arbitrary timeframes that differ by the factor of five.

First Screen

In an original version of the Triple Screen Trading System, the first screen involves trend-following indicators for the detection of trends on a weekly basis. Although trends can be detected by the classical MACD histogram or by the ADX directional system, we rather use the MACD indicator normalized by an Average True Range of the last n bars, where n is the length of the shorter moving average (Eq. 6.1).

$$\mathrm{MACDnorm}_{n,m}(i) = \frac{\mathrm{MACD}_{n,m}(i)}{\mathrm{ATR}_{n}(i)} \tag{6.1}$$

The normalized MACD returns zero if the market doesn't contain a trend, positive and negative numbers if the market is trending up and down, respectively.

The first screen serves as a filter which ensures that no position will be opened against the long-term trend. The slogan that characterizes the purpose of the first screen can be paraphrased as "*Trend is your friend; don't turn it into an enemy*".

We have outlined that Alexander Elder in the original version of the Triple Screen Trading System recommends using the MACD histogram to generate entry signals. A buy signal is generated just when the rising MACD histogram reaches its top and turns the direction. Similarly, sell signals are generated by the histogram bottoms.

Unfortunately, this definition is clear for a human trader, but not for a mechanical trading system. This is the reason why we use the normalized MACD rather than the original MACD histogram.

In fact, traders have three different options, to stay away, to buy and to sell. The first screen takes away one of them. The principle on which the first screen operates can be concluded into the two following rules:

1. IF *market is trending up* THEN *do not sell*
2. IF *market is trending down* THEN *do not buy*

Alexander Elder has proposed these rules for human traders. Conventional trading systems require these rules to be discretized into crisp thresholds. As we will see later, the trading systems based on fuzzy logic doesn't require such discretization.

Second Screen

The second screen is based on a timeframe five times shorter than the first screen. It detects opportunities in short-time swings of the market trend respecting the *buy low/sell high* paradigm. This means that the second screen generates buy signals when a market declines and sell signals when a market rises. Even though these signals are in the conflict with short-term movements, the first screen ensures that they always meet the long-term trend.

We we use the RSI oscillator to evaluate the short-term market balance. The corresponding rules for the second screen are then defined as follows:

1. IF *market is overbought* THEN *sell*
2. IF *market is oversold* THEN *buy*

Similarly to the previous case, a conventional trading system requires these rules to be discretized into the thresholds of 0.3 and 0.7 in order to meet the definition of RSI indicator described in Section 3.1.4.

Third Screen

The third screen is a discretionary technique for entering positions when decisions are actually made by the first and second screen. It works on a timeframe five times shorter than the second screen and doesn't generate any entry signals.

The principle of the third screen is based on trailing stop orders. When the first two screens generate a long entry signal, we place a trailing buy-stop order to catch the best possible buying opportunity. Similarly, the trailing sell-stops are used in cases of the short entry signals.

6.2.2 Fuzzy approach to the Triple Screen Trading System

Now, we will be about to implement fuzzy logic into a conventional Triple Screen Trading System in order to make it more robust and stable. The main benefit of the fuzzy approach is not that it improves system's performance, but that it allows us to bypass the optimization process.

First Screen

Due to several limitations of the trading simulator, strategies are not able to perform computations concurrently on multiple different timeframes.[32] There are two possible solutions of this problem. Although we could resample data window manually in the implementation of a trading strategy, the better solution is simply to multiple parameters of the corresponding oscillators. The signaling system of the first screen will be then constructed as follows:

$$trend = \mathrm{MACDnorm}_{n,m}(i) = \frac{\mathrm{MACD}_{n,m}(i)}{\mathrm{ATR}_{n}(i)}, \tag{6.2}$$

where parameters m and n are multiplied by the factor of five ($n = 12 \times 5 = 60$ and $m = 26 \times 5 = 130$).

[32] The ATS component of the coupled DEVS model contains only one data window (see Fig 4.8).

Next, we define a fuzzy linguistic variable *trend* which describes market's long-term behavior. This variable is defined in a range of -5 to 5 and contains five different linguistic terms with triangular membership functions:

Market trend (screen 1)	Min	Center	Max
μ_1^T - *Strong downtrend*	-5	-5	0
μ_2^T - *Slight downtrend*	-6	-3	0
μ_3^T - *No trend*	-4	0	4
μ_4^T - *Slight uptrend*	0	3	6
μ_5^T - *Strong uptrend*	0	5	5

These membership functions are graphically illustrated in Figure 6.2:

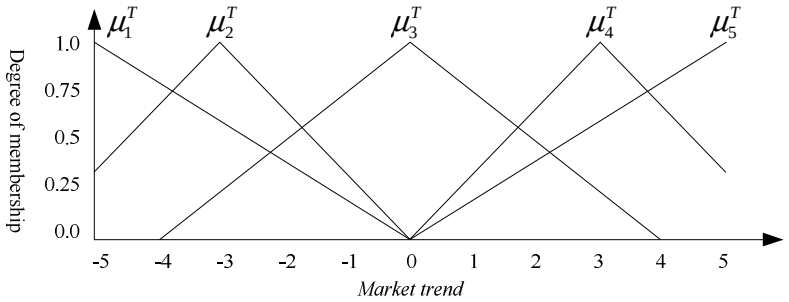


Figure 6.2: The fuzzy membership functions of the first screen of Triple Screen Trading System.

Second Screen

The second screen is implemented by the RSI oscillator with a period of twelve (Eq. 6.3). Since it operates on a timeframe five times shorter, we don't have to multiply its parameters:

$$balance = \mathrm{RSI}_n(i), \tag{6.3}$$

where $n = 12$.

This oscillator corresponds to the fuzzy linguistic variable *balance* with following membership functions:[33]

Market balance (screen 2)	Min	Center	Max
μ_1^B - *Very oversold*	-5	-5	0
μ_2^B - *Slightly oversold*	-6	-3	0
μ_3^B - *Balanced*	-4	0	4
μ_4^B - *Slightly overbought*	0	3	6

[33] Membership functions for the short-term market balance are the same as those on Figure 6.2.a in Section 6.2.

μ_5^B - *Very overbought*	0	5	5

Assessment of market situation

In the conventional Triple Screen Trading System, the first screen ensures that no position will be opened against the main trend and the second screen generates signals compatible with the main trend when a market is temporarily overbought or oversold. The purpose of a fuzzy inference mechanism is to evaluate how much a certain market situation is appropriate for entering the long or respectively the short position. In order to this, we define the output variable *assessment* in a range of -1 to 1 with the following membership functions:

Assessment	Min	Center	Max
μ_1^A - *Very good for selling*	-1	-1	-0.5
μ_2^A - *Good for selling*	-1	-0.5	0
μ_3^A - *Stay away*	-0.5	0	0.5
μ_4^A - *Good for buying*	0	0.5	1
μ_5^A - *Very good for buying*	0.5	1	1

Since the system which embeds the fuzzy-based trading strategy must receive a binary signal whether to or not to enter the position, the assessment of market situation must be discretized by a threshold for example as follows:

if assessment > 0.5 then buy
else if assessment < -0.5 then sell
else stay away

Money management

The attentive reader has probably noticed that we have been talking about the position entries, but not about the exits. The point is that the screens of the Triple Screen Trading System don't define when to exit the position. Because of this, we use a simple money management technique based on moving stop-losses. Considering that the first screen of the trading system is based on a trend-following paradigm, we don't recommend cutting profits by profit targets.

Our money management is based on two rules. At first, we must assure that at every moment at least 60% of the profit excursion[34] will be secured by a stop-loss.

After that, we place an initial stop-loss several ticks above/below the entry price. The number of ticks also depends on market conditions. Intuitively, if a market is extremely oversold or overbought, we don't expect that it has further power to extend such irregularity. Because of this, we use tight stop-losses in such cases.

The power of trend is another considerable aspect of the market behavior. Strong and fast moving trends usually involve also bigger retracements which may cause accidental execution of tight stop-losses. Hence, in such cases, we usually post wider stop-losses.

Let's define the linguistic variable *stop-loss* in a range of \$2 to \$6.[35] This variable contains three different linguistic terms – *tight*, *normal*, and *wide* as shown in Figure 6.3.

[34] The profit excursion represents the largest hypothetical profit earned by a single trade while it is open.

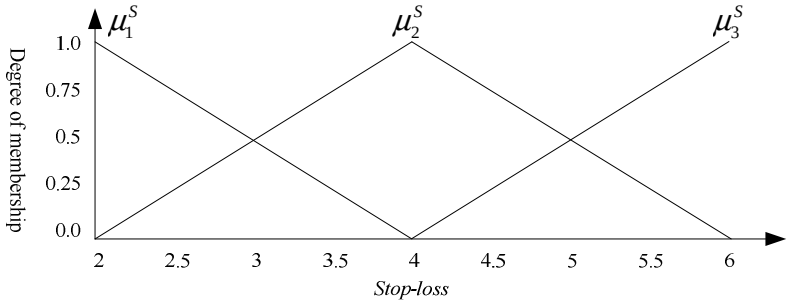


Figure 6.3: The *stop-loss* linguistic variable of the Triple Screen Trading System.

The triangular membership functions are numerically defined in the following table:

Stop-loss	Min	Center	Max
μ_1^S *- Tight stop-loss*	2	2	4
μ_2^S *- Normal stop-loss*	2	4	6
μ_3^S *- Wide stop-loss*	4	6	6

Inference mechanism and fuzzy rules

Our inference mechanism maps two inputs to the two outputs, as it is illustrated in Figure 6.4. Each time when the strategy receives a new quote, it recalculates indicators for both screens and pushes them into the fuzzy controller. The fuzzy controller assesses market situation, computes stop-losses and returns these values back to the strategy.

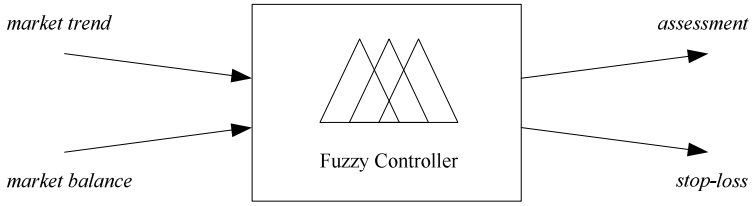


Figure 6.4: The fuzzy controller of the Triple Screen Trading System.

The mapping between the antecedents and consequents is performed by rules in the form:

if trend is <?> *and balance is* <?> *then assessment is* <?> *and stop-loss is* <?>

Considering that these rules cover almost all combinations of linguistic terms, we write them in the form of a table. The following table contains rules for the assessment of market situation.

Assessment					
Balance / Trend	Very oversold	Slightly oversold	balanced	Slightly overbought	Very overbought

[35] The distance between the entry price and the stop-loss depends on a used strategy, market, timeframe and other aspects.

Strong downtrend	Stay away	Stay away	Good for selling	Very good for selling	Very good for selling
Slight downtrend	Stay away	Stay away	Good for selling	Good for selling	Very good for selling
No trend	Stay away	Stay away	Stay away	Stay away	Stay away
Slight uptrend	Very good for buying	Good for buying	Good for buying	Stay away	Stay away
Strong uptrend	Very good for buying	Very good for buying	Good for buying	Stay away	Stay away

The width of stop-losses is determined by the following mapping:

Stop-loss					
Balance / Trend	Very oversold	Slightly oversold	balanced	slightly overbought	Very overbought
Strong downtrend	Normal stop-loss		Wide stop-loss		Normal stop-loss
Slight downtrend		Normal stop-loss		Normal stop-loss	
No trend	Tight stop-loss		Normal stop-loss		Tight stop-loss
Slight uptrend		Normal stop-loss		Normal stop-loss	
Strong uptrend	Normal stop-loss		Wide stop-loss		Normal stop-loss

Even though we have used two independent tables, the consequents in corresponding cells are joined together into the same rule by the logical *and*. The fuzzy controller is nothing more than a function constructed according to these tables that maps two inputs to two outputs. This function can be graphically illustrated by a fuzzy surface, as shown in Figure 6.5. As we can see, the shape of these surfaces directly corresponds to the content of the rule tables.

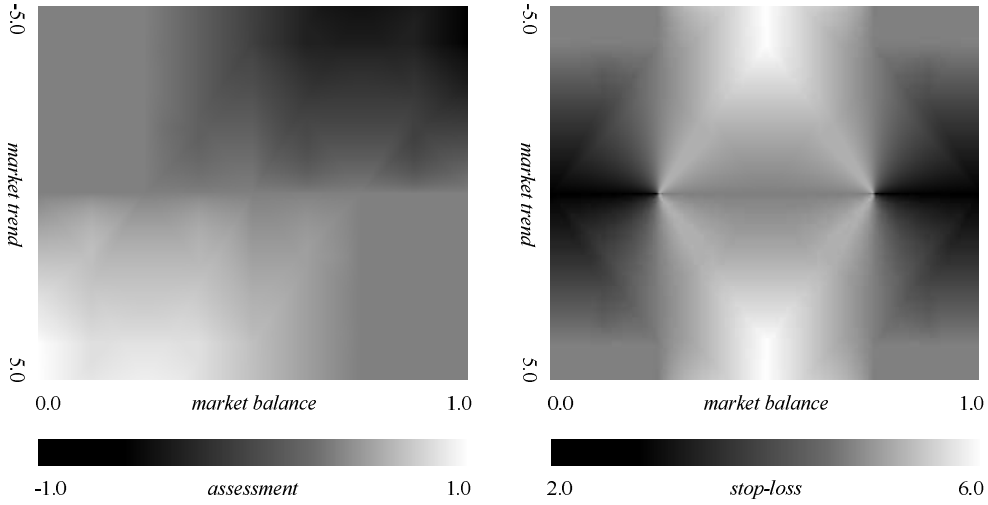


Figure 6.5: Fuzzy surfaces for the output linguistic variables *assessment* and *stop-loss*.

6.3 Analysis of the sensitivity and robustness

The purpose of this section is to evaluate how much the fuzzy approach increases robustness of conventional trading strategies. The robustness can be valuated by a sensitivity analysis that studies how much a change in parameters of a trading strategy affects its overall performance. The other way is to study how much the signaling system of a strategy is sensitive to changes in market conditions.

6.3.1 Sensitivity analysis of the whole market system

If we assume that a simulator with an embedded parametric trading strategy and market data is also a system, we can study how much a change in the parameter p_i affects the overall performance (fitness) of the trading strategy on given market data (Eq. 6.4).

$$S_{f,p_i} = \left.\frac{\partial f}{\partial p_i}\right|_{p_{i0}}, \tag{6.4}$$

where f is a fitness function of the parametric trading strategy, p_i is the i^{th} parameter and S_{f,p_i} represents a sensitivity of the strategy to changes in this parameter.

Although this concept theoretically seems very good, it is practically inapplicable to most of the strategies including the Triple Screen Trading System doe to several reasons. At first, the parameter space of many of the trading strategies is not continuous and not sufficiently wide.

Secondly, the sensitivity analysis depends also on the given market data which makes it very unreliable for different market conditions. It is practically impossible to find two different market situations on which the sensitivity analysis of the same strategy produces the same results.

Because of this, this type of analysis is not able to provide us a comprehensive description of the strategy robustness regardless of the market data. This is the reason why the stability of trading strategies cannot be studied in such way.

6.3.2 Sensitivity analysis of the signaling system

Considering that the sensitivity analysis of the whole market system is unreliable doe to different behavior in different market conditions, we will analyze only the sensitivity of a signaling system.

The conventional version of the Triple Screen Trading System implements a signaling system only by using two strict thresholds for the market trend and market balance (as we have explained in Section 6.2.1). If these thresholds are met, the signaling system produces a crisp signal “1” for long positions and “-1” for short positions. Otherwise, it produces a zero.

The sensitivity of the signaling system to changes in a particular input is defined by the partial derivative of the system’s output (assessment) with respect to such input (Eq. 6.5).

$$S_{a,t} = \left.\frac{\partial a}{\partial t}\right|_{t_0} \; ; \; S_{a,b} = \left.\frac{\partial a}{\partial b}\right|_{b_0}, \tag{6.5}$$

where a is the assessment produced by a signaling system, t is the valuation of market trend (screen 1) and b is the valuation of market's short-term balance (screen 2). The sensitivity to all inputs is then given as a gradient of the fuzzy controller's output (Eq. 6.6):

$$S_a = \left(\frac{\partial a}{\partial t}, \frac{\partial a}{\partial b} \right) \Bigg|_{t_0, b_0} \tag{6.6}$$

The sensitivity S_a can be graphically illustrated for all combinations of input values by a so-called sensitivity surface.

Conventional Triple Screen Trading System

The crisp thresholds of the conventional Triple Screen Trading System may cause that its response will be very sensitive to input changes in several points. For the indicator of market trend, the system's sensitivity goes to the infinity for values of 2.0 and -2.0. Similarly, the signaling system is very sensitive for values 0.3 and 0.7 of the short-term market balance.

The maximum sensitivity of the conventional Triple Screen Trading System for all combinations of input values is then given as follows (Eq. 6.7):

$$S_a' = \max_{t \in T, b \in B} \left\{ \left| S_a(t, b) \right| \right\} = \infty \tag{6.7}$$

The infinite sensitivity is common for all conventional trading systems which are based on hard constraints and crisp what-if rules. We are convinced that it is very important to keep the maximum sensitivity S_a' in a limited range in order to preserve robustness and stability of the whole trading system.

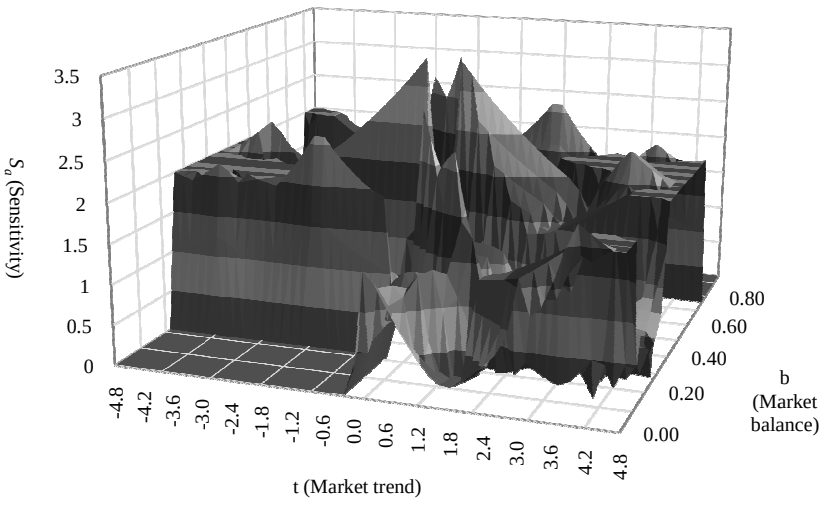


Figure 6.6: The sensitivity surface of the fuzzy-based Triple Screen Trading System.

Fuzzy-based Triple Screen Trading System

Now, we take a look to the sensitivity of the fuzzy-based Triple Screen Trading System. This system implements a fuzzy controller that replaces crisp what-if rules. Thanks to this, the controller has a limited sensitivity to the changes in its inputs. This sensitivity is graphically illustrated by the sensitivity surface in Figure 6.6.
As we can see in Figure 6.6, the overall sensitivity of the fuzzy controller is restricted from the top. This restriction is given as:

$$S_a' = \max_{t \in T, b \in B} \left\{ \left| S_a(t,b) \right| \right\} = 3.36 \tag{6.8}$$

The restriction of 3.36 is also a maximum gradient of the fuzzy surface. It assures that a limited change in input parameters causes a limited change in a response of the signaling system. The ratio between these changes is just a 3.36.

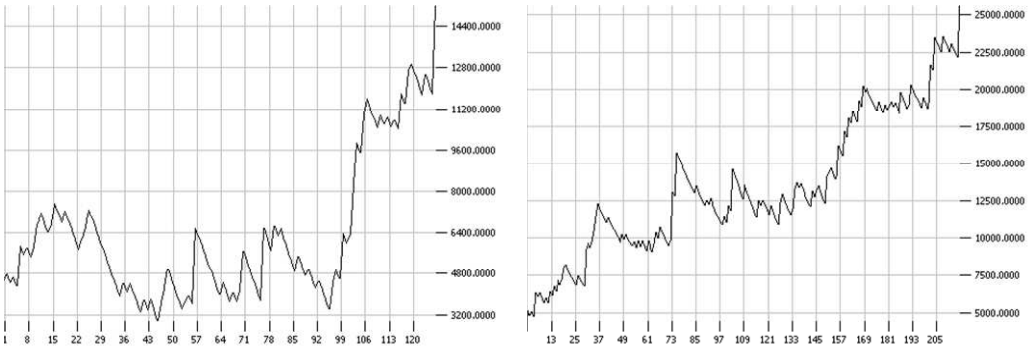
The unlimited sensitivity is an unwanted property for each signaling system. However, this doesn't necessarily mean that lesser sensitivity is better. Someone might say that more sensitive signaling systems are appropriate for more volatile markets, but we don't think so. Although this proportion surely depends on market conditions, we are not able to recommend to readers any general patterns or rules.

6.4 Case study

In this case study, we run both conventional and fuzzy-based triple screen trading systems on historical market data. Both systems are constructed as shown in sections 6.2.1 and 6.2.2. We will trade CFDs of the Russell 2000 Index between years 1991 and 2009 on a daily chart. Since this index measures the performance of the whole small-cap segment of the United States equities, it has relatively stable behavior, appropriate for the automated trading. We also assume that a broker requires a margin of 5% for each CFD contract. For a better comparison of both strategies, we won't apply the position sizing.

At first we run the conventional Triple Screen Trading System on the Russell 2000 Index with the starting capital of $4500. The parameters of such system are chosen ad-hoc without any optimization. This is very important, since we have an assurance that the corresponding outcome won't be the product of a nasty curve-fitting.

The conventional strategy proposed in Section 6.2.1 employs Normalized MACD indicator with the periods of 12×5 and 26×5. The period of the RSI indicator is twelve. The strategy generates buy and sell signals if the first screen overcomes levels of 2.0 and -2.0, respectively. These signals are confirmed by the second screen with the RSI levels of 0.3 and 0.7. The equity curve that corresponds to this strategy is shown in Figure 6.7.a.



A B **Figure 6.7:** Equity curves of the conventional **(a)** and fuzzy-based **(b)** triple screen trading systems.

Secondly, we apply the fuzzy-based Triple Screen Trading System exactly as it has been proposed in Section 6.2.2. This system uses the same periods of indicators as the conventional one, but it employs fuzzy logic instead of crisp thresholds for the first and second screen.

Figure 6.7.b contains an equity curve that illustrates successfulness of the fuzzy-based Triple Screen Trading System. Additionally, the following table compares statistics of both systems:

Conventional Triple Screen Trading System		Fuzzy-based Triple Screen Trading System	
Start	1991-04-30 13:30:00	Start	1991-03-20 13:30:00
End	2009-01-28 15:57:59	End	2009-01-28 15:57:59
Number of trades	126	Number of trades	216
Number of wins	49	Number of wins	76
Number of losses	77	Number of losses	140
Number of profit target exits	0	Number of profit target exits	0
Number of stop-loss exits	126	Number of stop-loss exits	216
Number of strategy exits	0	Number of strategy exits	0
Average number of stop-loss moves	1.75	Average number of stop-loss moves	1.67
Max contracts per trade	75	Max contracts per trade	75
Min contracts per trade	75	Min contracts per trade	75
Avg contracts per trade	75.00	Avg contracts per trade	75.00
Equity start	4500.00	Equity start	4500.00
Equity high	15203.12	Equity high	25564.16
Equity low	2983.94	Equity low	4500.00
Final equity	15203.12	Final equity	25564.16
Return of starting equity	237.85%	Return of starting equity	468.09%
Largest win	3450.15	Largest win	3450.15
Largest loss	-501.11	Largest loss	-451.61
Average win	822.03	Average win	888.67
Average loss	-384.11	Average loss	-331.96
Max consecutive wins	3	Max consecutive wins	4
Max consecutive losses	9	Max consecutive losses	9
Average profit excursion	660.29	Average profit excursion	639.66
Average adverse excursion	-280.92	Average adverse excursion	-241.86
Max profit excursion	5929.50	Max profit excursion	5929.50
Max adverse excursion	-501.11	Max adverse excursion	-451.61
Max drawdown	4533.77	Max drawdown	4908.49

Max position duration	4 mns 13 dys	Max position duration	5 mns 21 dys
Min position duration	18 min 47 sec	Min position duration	13 min 23 sec
Avg position duration	9 dys 21 hrs	Avg position duration	8 dys 28 hrs
Annual Return	7.09%	Annual Return	10.21%

The statistics reveal that the fuzzy approach brings much more stable behavior than conventional techniques. Although the maximum account drawdown is in both cases roughly equal, profits of the fuzzy-based trading system are distributed more uniformly along the equity curve.

The final equity of the conventional method is a little over $15000, whereas the fuzzy approach boosted it up to $25564. However, it is not statistically proven that the fuzzy approach generally increases overall outcomes of trading strategies.

The fuzzy approach makes trading strategies more robust and equity curves more regular. It also limits sensitivity of trading strategies by eliminating crisp decisions in the underlying signaling systems. Another considerable advantage is also the fact that fuzzy approach allows us to bypass the optimization process.

Summary

One of the most essential objectives of this thesis was to comprehensively research all possible topics in which the various intelligent systems may be involved to analyze, assess or predict market behavior. Considering that the construction of one big, sophisticated and single-purpose automated trading system wouldn't cover all these topics, this work deals with particular problems separately in several individual chapters.

Processes on financial markets can be researched and described generally in two different manners. The approach oriented to a *structural* description of market behavior studies the behavior of individual agents acting on financial markets in terms of the Kahneman and Tversky's Prospect Theory as well as the synergic effect of the whole market from the system theory point of view. Theoretically, the structural description is able to answer the question *why* markets act as they do.

We share the assumption that the multi-agent system which is perfectly modeled in terms of the prospect theory results into the same behavior as it is described in the Elliott's Wave Principle. In other words, the prospect theory is one of the pylons that support Elliott's behavioral description of market movements. If we compare the theories introduced in the first chapter, we find that both the efficient market hypothesis and behavioral market theory supports this assumption, but the chaos theory states that neither structural nor behavioral description of market movements can give us a respectable outcome.

The problem is that it is practically impossible to structurally model behavior of the whole market. Although the second chapter explains several common principles of the system theory related to self-organizing hierarchical systems, this work as a whole doesn't have an ambition to model markets structurally. However, this topic would be a perspective subject of the further research.

This thesis is intended mainly to a *behavioral* description of market movements. This approach is supported by the Elliott wave principle explained in the second chapter and by other technical analysis methods presented in Chapter Three. Generally, the whole technical analysis is based on a behavioral description of market movements. This type of description tells us *how* market moves, but not *why* it moves as it do.

Considering that we are not able to perform simulations of the market behavior, we simulate the behavior of trading systems on known market movements. These systems usually implement simple mechanical strategies based on a hard computing paradigm. This thesis brings alternative, soft computing computational models to trading strategies and innovatively combines two different areas of science – the artificial intelligence and the technical analysis. One of the main benefits of this work is a demonstration that the soft computing approach in a combination with the "soft" social sciences accounts more reliable results than the conventional mathematical models.

Bibliography and further reading

1. **Mullainathan, Sendhil and Thaler, Richard H.** Behavioral Economics. *International Encyclopedia of the Social & Behavioral Sciences.* 2001. pp. 1094-1100. ISBN 978-0-08-043076-8.

2. **Plummer, Tony.** *Forecasting Financial Markets.* London, UK : Kogan Page Ltd, 2003. ISBN 978-0-74-943939-2.

3. **Kahneman, Daniel and Tversky, Amos.** Prospect Theory: An Analysis of Decision under Risk. *Econometrica.* s.l. : The Econometric Society, 1979. Vol. 47, pp. 263-291.

4. **Jordan, Dominic W. and Smith, Peter.** *Nonlinear Ordinary Differential Equations - An Introduction to Dynamical Systems.* s.l. : Oxford University Press, 1999. ISBN 978-0-19-856562-8.

5. **Frost, Alfred J. and Prechter, Robert R.** *Elliott Wave Principle - Key to Market Behavior.* s.l. : Elliott Wave International Inc, 1998. ISBN 978-0-93-275043-3.

6. **Graham, Benjamin and Zweig, Jason.** *The Intelligent Investor.* New York, USA : HarperCollins, 2003. ISBN 978-0-06-055566-5.

7. **Pezzutti, Paolo.** *Trading the US Markets: A Comprehensive Guide to Us Markets for European Traders and Investors.* s.l. : Harriman House Limited, 2008. ISBN 978-1-90-564105-5.

8. **Schwartz, Robert A. and Francioni, Reto.** *Equity Markets in Action: The Fundamentals of Liquidity, Market Structure & Trading.* s.l. : John Wiley and Sons, 2004. ISBN 978-0-47-146922-3.

9. **Velez, Oliver and Capra, Greg.** *Tools and Tactics for the Master Day Trader: Battle-Tested Techniques for Day, Swing, and Position Traders.* s.l. : McGraw-Hill Professional, 2000. ISBN 978-0-07-136053-1.

10. **LaBier, Rogan M.** *The Nasdaq Trader's Toolkit.* s.l. : John Wiley and Sons, 2001. ISBN 978-0-47-140403-3.

11. **Harris, Larry E.** *Trading and Exchanges: Market Microstructure for Practitioners.* s.l. : Oxford University Press, 2003. ISBN 978-0-19-514470-3.

12. **Blum, Avrim, Sandholm, Tuomas and Zinkevich, Martin.** Online algorithms for market clearing. *Journal of the ACM.* New York : ACM, September 2006. Vol. 53, 5, pp. 845-879. ISSN 0004-5411.

13. **Trevathan, Jarrod and Read, Wayne.** Variable Quantity Market Clearing Algorithms. *E-Business and Telecommunication Networks.* s.l. : Springer Berlin Heidelberg, 2006, Vol. IX, pp. 28-39.

14. **Elder, Alexander.** *Trading for a Living.* s.l. : John Wiley and Sons, 1993. ISBN 978-0-47-159224-2.

15. **Zeigler, Bernard P.** *On the Feedback Complexity of Automata.* s.l. : Management Information Services, 1970.

16. **Zeigler, Bernard P., Praehofer, Herbert and Kim, Tag G.** *Theory of Modeling and Simulation.* s.l. : Academic Press, 2000. ISBN 978-0-12-778455-7.

17. **Haupt, Randy L. and Haupt, Sue E.** *Practical Genetic Algorithms.* s.l. : Wiley-IEEE, 2004. ISBN 978-0-47-145565-3.

18. **Zadeh, Lotfi A.** Roles of Soft Computing and Fuzzy Logic in the Conception, Design and Deployment of Information/Intelligent Systems. *Computational Intelligence: Soft Computing and Fuzzy-neuro Integration with Applications.* s.l. : Springer, 1998. ISBN 978-3-54-064004-2.

19. **Klir, George J. and Yuan, Bo.** *Fuzzy Sets and Fuzzy Logic: Theory and Applications.* s.l. : Prentice Hall PTR, 1995. ISBN 978-0-13-101171-7.

20. **Babcock, Bruce.** Chaos Theory and Market Reality. [Online] Reality Based Trading Company. http://www.rb-trading.com/article4.html.

21. **Neely, Glenn and Hall, Eric.** *Mastering Elliott Wave.* s.l. : Windsor Books, 1990. ISBN 978-0-93-023344-0.

22. **Prechter, Robert R.** *The Major Works of R. N. Elliott.* s.l. : New Classics Library, 1990. ISBN 978-0-93-275015-0.

23. **Peters, Edgar E.** *Fractal Market Analysis: Applying Chaos Theory to Investment and Economics.* s.l. : John Wiley and Sons, 1994. ISBN 978-0-471-58524-4.

24. **Fishwick, Paul A.** *Simulation Model Design and Execution.* s.l. : Prentice Hall, 1995. ISBN 978-0-13-098609-2.

25. **Kirkpatrick, Charles D. and Dahlquist, Julie R.** *Technical Analysis: The Complete Resource for Financial Market Technicians.* s.l. : FT Press, 1996. ISBN 978-0-13-153113-0.

26. **Kiel, Douglas L. and Elliott, Euel W.** *Chaos Theory in the Social Sciences: Foundations and Applications.* s.l. : University of Michigan Press, 1997. ISBN 978-0-47-208472-2.

27. **Katz, Jeffrey O. and McCormick, Donna.** *The Encyclopedia of Trading Strategies.* s.l. : McGraw-Hill, 2000. ISBN 978-0-07-058099-2.

28. **Lipton, Alexander.** *Mathematical methods for foreign exchange: a financial engineer's approach.* s.l. : World Scientific, 2001. ISBN 978-9-81-024615-0.

29. **Russel, Stuart J., Norvig, Peter and Canny, John F.** *Artificial Intelligence, a Modern Approach.* s.l. : Pearson Education Inc., 2003. ISBN 978-0-13-790395-5.

30. **Camerer, Colin, Loewenstein, George and Rabin, Matthew.** *Advances in Behavioral Economics.* s.l. : Princeton University Press, 2004. ISBN 978-0-69-111681-5.

31. **Weissman, Richard L.** *Mechanical Trading Systems: Pairing Trader Psychology with Technical Analysis.* s.l. : John Wiley and Sons, 2004. ISBN 978-0-47-173097-2.

32. **Kaufman, Perry J.** *New Trading Systems and Methods.* s.l. : John Wiley and Sons, 2005. ISBN 978-0-47-126847-5.

33. **Smithson, Michael and Verkuilen, Jay.** *Fuzzy Set Theory: Applications in the Social Sciences.* s.l. : SAGE, 2006. ISBN 978-0-76-192986-4.

34. **McElreath, Richard and Boyd, Robert.** *Mathematical Models of Social Evolution.* s.l. : University Of Chicago Press, 2007. ISBN 978-0-22-655827-1.

Notations, functions and mathematical symbols

Charts

o_i	Opening price of the i^{th} bar.
h_i	Highest price of the i^{th} bar.
l_i	Lowest price of the i^{th} bar.
c_i	Closing price of the i^{th} bar.
v_i	Volume of the i^{th} bar.

Indicators and oscillators

FER_n	Fractal efficiency ratio with a period of n.
$\text{MA}_n(i)$	Simple Moving Average of the i^{th} bar (with a period of n).
$\text{EMA}_n(i)$	Exponential Moving Average of the i^{th} bar (with a period of n).
$\text{MACD}_{\text{n,m}}(i)$	Moving Average Convergence / Divergence indicator of the i^{th} bar based on exponential moving averages with the periods of m and n.
$\text{DM}(i)$	Directional movement of the i^{th} bar.
$\text{TR}(i)$	True range of the i^{th} bar.
$\text{ADX}(i)$	Average Directional Index of the i^{th} bar.
$\text{ATR}_n(i)$	Average True Range of the i^{th} bar (with a period of n).
$\text{RSI}_n(i)$	Relative Strength Index of the i^{th} bar (with a period of n).
$\%\text{B}_{n,k}(i)$	Bollinger %B indicator of the i^{th} bar.

Trade statistics

ntr	Number of trades	*lwi*	Largest win
nwi	Number of wins	*llo*	Largest loss
nlo	Number of losses	*awi*	Average win
npt	Number of PT exits	*alo*	Average loss
nsl	Number of SL exits	*maxcw*	Max consecutive wins
nsx	Number of SX exits	*maxcl*	Max consecutive losses
aslm	Avg. number of SL moves	*avgpe*	Average profit excursion
maxc	Max contracts per trade	*avgae*	Average adverse excursion
minc	Min contracts per trade	*maxpe*	Max profit excursion
avgc	Avg. contracts per trade	*maxae*	Max adverse excursion
est	Equity start	*maxad*	Max account drawdown
ehi	Equity high	*maxpd*	Max position duration
elo	Equity low	*minpd*	Min position duration
efi	Equity final	*avgpd*	Avg. position duration
req	Return on equity	*areq*	Annual return on equity

Sets and sequences

$\left[x_j \right]_{j=0}^{n-1}$	The series of elements $x_0, x_1, \ldots, x_{n-1}$.
$\left\{ x_j \right\}_{j=0}^{n-1}$	The set of elements $x_0, x_1, \ldots, x_{n-1}$.
mean A	Mean value of the A, where A can be the set or the sequence.
std A	Standard deviation of the set or sequence A.
max A	Maximum value contained in the set or sequence A.
min A	Minimum value contained in the set or sequence A.
$\lvert A \rvert$	The cardinality of the set A (Number of elements contained in the set).

Pseudo-code symbols

$c \rightarrow v$	Pointer operator, denotes variable (or method) v of the component c.

Others

φ	The Golden ratio.

Appendix A

Implementation overview

All principles described in this work were researched and tested on a framework for composing and analysis of trading strategies. This framework is implemented as a project in the Java™ programming language and contains various simulation and optimization tools as well as soft computing tools for embedding fuzzy logic, genetic algorithms and neural networks to trading strategies. Additionally, it provides also a GUI kit for a better and more comfortable presentation of simulation outputs. Since all market data is stored in a SQL database, the framework contains tools for its automatic actualization and synchronization.

This appendix is not a comprehensive reference guide to the trading framework. Its purpose is only to provide a brief overview of the system at a glance. Because of this, it is organized chronologically as a tutorial in order to make readers familiar with basic principles of the framework.

A.1 Market data

The OHLC market data is organized as time series of prices. Each series is based on a certain timeframe and corresponds to a certain security traded on a certain market. This hierarchy is reflected also in tables of the relational database.

As we can see in Figure A.1, the database contains four different tables – *market*, *symbol*, *chart* and *quote*. Each financial market (exchange) contains an identifier (primary key) and several other properties, such as the name, location and the time zone. Traded securities (instruments) are represented by the *symbol* table, which contains symbol identifier (`market`, `symbol`), symbol name and type of the instrument. Each security may be associated to several charts, each with a different timeframe. Because of this, charts are identified by a symbol identifier (`market`, `symbol`) and by the `timeframe`. Finally, a chart is represented by an ordered sequence of OHLC quotes. Quotes are identified by the chart identifier (`market`, `symbol`, `timeframe`) and by the underlying time period (`start`, `end`).

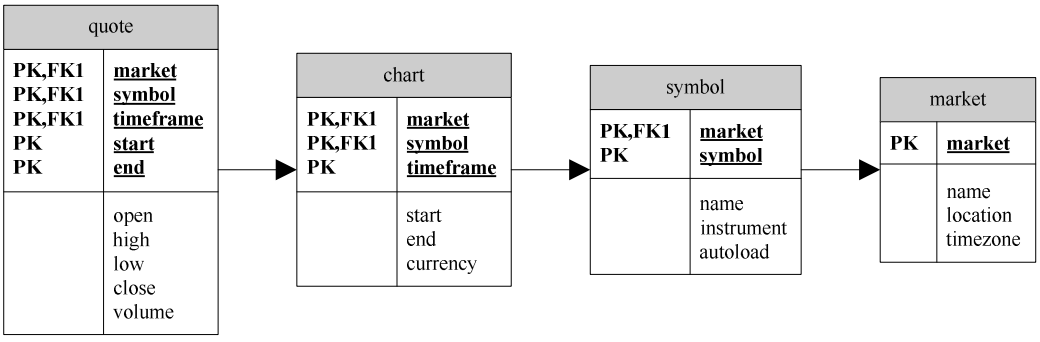


Figure A.1: Structure of market data in the relational database.

Although the framework is able to connect to an arbitrary database management system via the JDBC interface, the market data is provided in a form of the MySQL™ dump file. The database contains almost six thousand securities from forty-one markets with the aggregated size of nearly 2.2 GB.

The framework implicitly looks for SQL credentials in the `dbconfig.xml` file. If the file doesn't contain proper values or if the provided values result into an unsuccessful attempt, the framework displays an interactive dialog prompt.

A.2 Features of the framework

The trading framework is organized as a hierarchy of Java™ packages. These packages implement tools and widgets for connecting to a database server, charting and GUI tools, trading simulator and trading analyzer, various optimization tools as well as fuzzy and neural-network toolboxes.

Considering that the whole framework contains nearly eighty packages with dozens of classes, we state only the major ones.

Package ats.database

This package handles the connection to a database server, implements classes that wrap JDBC connector and provides application functionality for retrieving and storing persistent data.

Since the time series of market data may be excessively long, it is not suitable to retrieve them at once by a single query. Because of this, the package `ats.database` contains also a buffered reader which is able to read data from the database step by step.

In the implementation, a price chart (as a time series) is stored in an object of the `Chart` class with the `QuoteSequence` interface, as shown in Code A.1.

Code A.1:

```
public interface QuoteSequence {
   public int getLength();
   public Quote getQuoteAt(int index);
   public Quote getQuoteFromEnd(int index);
   public Quote getFirstQuote();
   public Quote getLastQuote();
}
```

Quotes from the chart can be retrieved using the `getQuoteAt` method. Each quote contains underlying OHLC values and information about the traded volume.

Package ats.sync

The package `ats.sync` contains synchronization tools which keep the database up to date. Quotes are fetched from various public sources via the http protocol. The synchronization process can be executed by invoking `ats.sync.Sync` class.

Package ats.gui

The package `ats.gui` is intended to support graphical user interface and visualization tools for other components of the trading framework. One of the main features of this package is an

interactive charting tool which is able to graphically display price charts, indicators, oscillators as well as the trading orders.

At first, we display the chart selection dialog box by invoking `ats.gui.ChartSelection` class. The selection box scans the database for all symbols and displays corresponding drop-down lists (Fig. A.2).

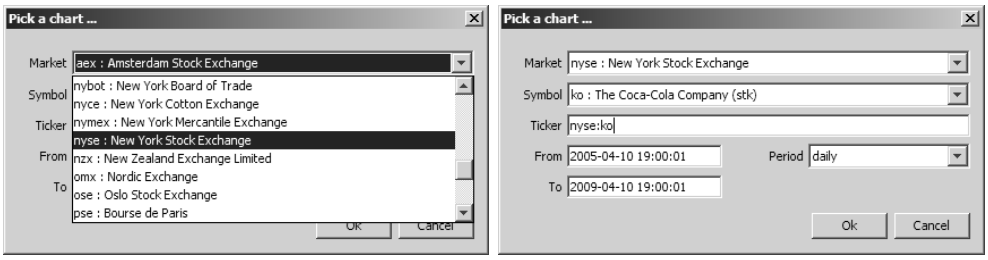


Figure A.2: The chart selection dialog box.

Figure A.3 contains a charting window for the common stocks of the Coca-Cola Company. The chart can be arbitrarily scrolled, zoomed or scaled. Additionally, the interactive cursor always provides appropriate details about the underlying bar.

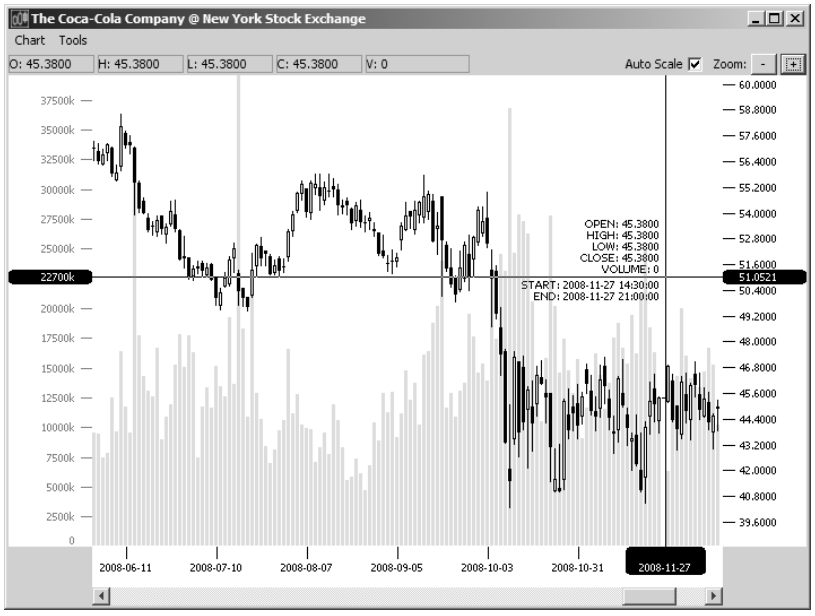


Figure A.3: The interactive charting tool of the trading framework.

Now, let's take a look to the development environment for the analysis of trading strategies. The environment can be invoked by the execution of `ats.gui.strategyIde.Ide` class. This action displays the window shown in Figure A.4.

At first, we create a new trading strategy or open an existing one (*Strategy* > *Open*). After that, we select the simulation data (*Market data* > *Select from database*). The active market data are then indicated by the text box in the bottom right corner of the IDE window.

When both trading strategy and market data is defined, we run the simulation by clicking to the *Run* button near the status pane. This action compiles the opened trading strategy and runs the simulation on the given market data.

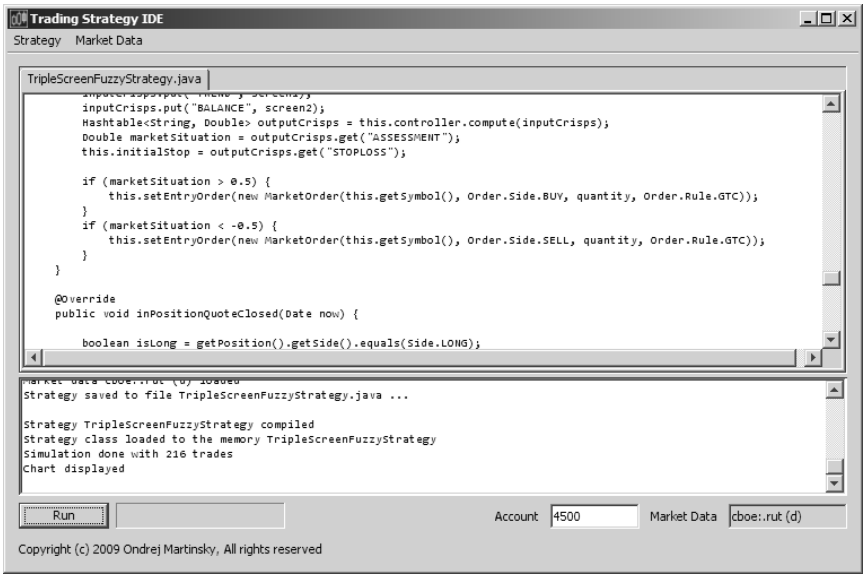


Figure A.4: The integrated development environment for the analysis of trading strategies.

The simulation engine automatically opens several windows with the results. Generated orders can be displayed directly in the interactive chart as shown in Figure A.5.

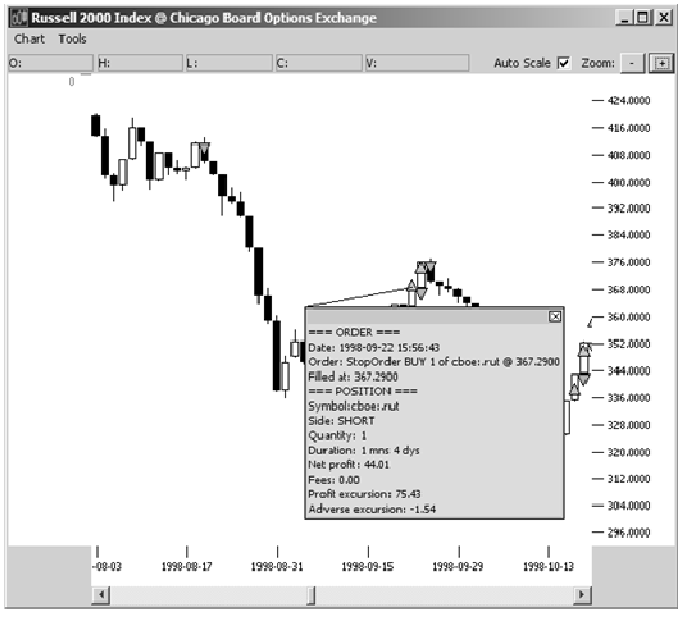
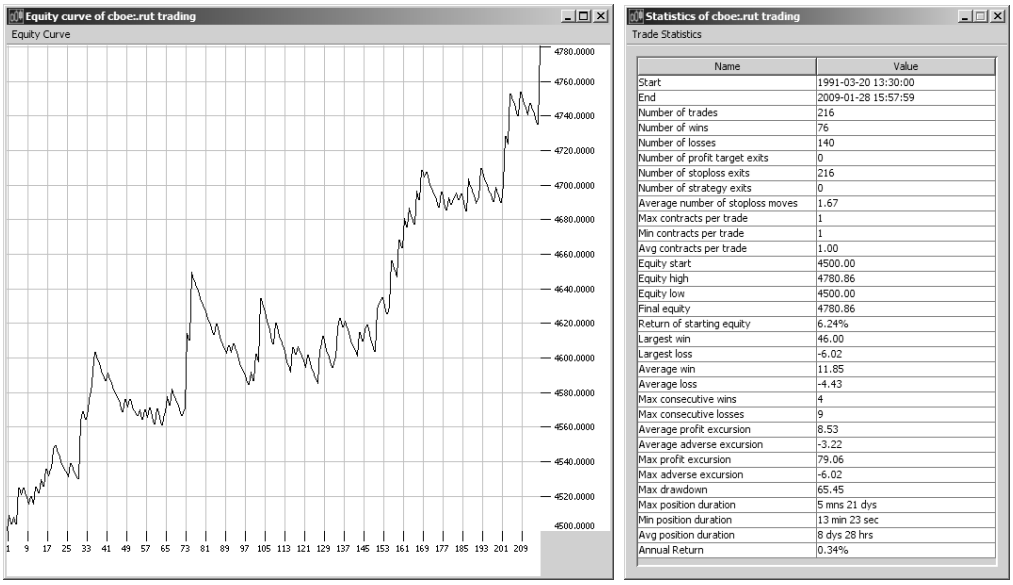


Figure A.5: The interactive chart displays also trading orders which have been generated during the simulation.

The second option is to display the corresponding equity curve (*Tools* > *Equity Curve*) as shown in Figure A.6.a. An equity curve is often used to illustrate the chronological record of all profits and losses.

The simulator provides also the statistics of trading. This statistics is accessible via the *Tools* > *Statistics* menu command, as shown in Figure A.6.b.



Name	Value
Start	1991-03-20 13:30:00
End	2009-01-28 15:57:59
Number of trades	216
Number of wins	76
Number of losses	140
Number of profit target exits	0
Number of stoploss exits	216
Number of strategy exits	0
Average number of stoploss moves	1.67
Max contracts per trade	1
Min contracts per trade	1
Avg contracts per trade	1.00
Equity start	4500.00
Equity high	4780.86
Equity low	4500.00
Final equity	4780.86
Return of starting equity	6.24%
Largest win	46.00
Largest loss	-6.02
Average win	11.85
Average loss	-4.43
Max consecutive wins	4
Max consecutive losses	9
Average profit excursion	8.53
Average adverse excursion	-3.22
Max profit excursion	79.06
Max adverse excursion	-6.02
Max drawdown	65.45
Max position duration	5 mns 21 dys
Min position duration	13 min 23 sec
Avg position duration	8 dys 28 hrs
Annual Return	0.34%

Figure A.6: (a) The equity curve represents an impact of the trading strategy to trading account. **(b)** The statistics of simulated trading.

The package `ats.gui` contains also tools for visual analysis of Elliott waves. Because of this, the interactive charting window is able to display detected hierarchical Elliott wave (Fig. A.7).

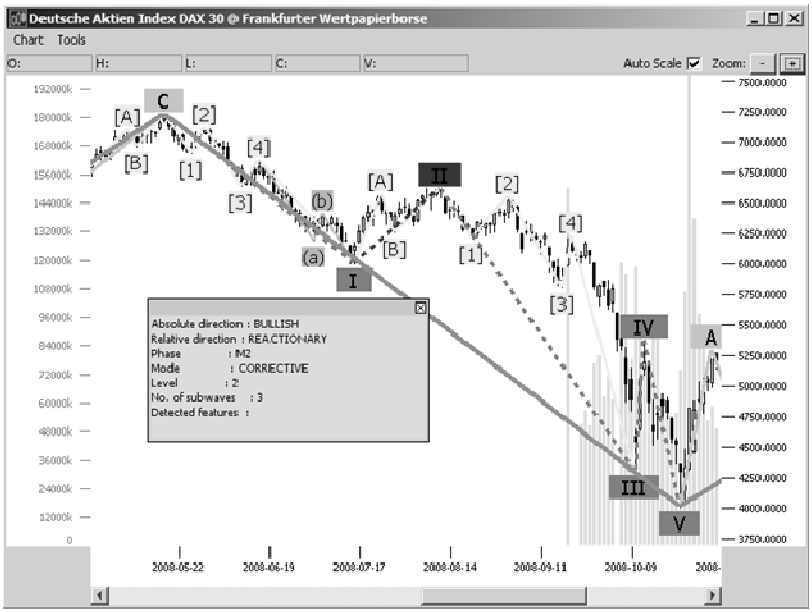


Figure A.7: The visualization of the detected Elliott wave.

Package ats.strategy

Package `ats.strategy` provides interfaces for both conventional and parametric trading strategies. These strategies are based on the same principle as it has been presented in Section 4.7 of this thesis. The interface for conventional trading strategies is shown in Code A.2.

Code A.2:

```
public interface Strategy {
   public void set(Environment environment,
                   QuoteWindow window,
                   ConfigBrokerIF broker,
                   PositionController position,
                   AtsComponent atsComponent);
   public void clear();
   public void outOfPositionQuoteOpened(Date now, Double openPrice);
   public void outOfPositionQuoteClosed(Date now);
   public void positionEntered(Date now);
   public void inPositionQuoteOpened(Date now, Double openPrice);
   public void inPositionQuoteClosed(Date now);
   public void outOfPositionMarketOpened(Date now, Double openPrice);
   public void outOfPositionMarketIsClosing(Date now);
   public void inPositionMarketOpened(Date now, Double openPrice);
   public void inPositionMarketIsClosing(Date now);
   public int getWindowCapacity();
   public Object clone();
}
```

In addition, parametric trading strategies contain methods for storing and retrieving parameters (see Code A.3). These parameters are later used by various optimization processes.

Code A.3:

```
public interface ParametricStrategy extends Strategy {
   public void setParameters(ParameterSet parameters);
   public ParameterSet getParameters();
   public void initParameters();
}
```

Package ats.neuralNetwork

Package `ats.neuralNetwork` contains an implementation of the back-propagation neural network. Since neural networks have practical application in a wide range of classification and optimization tasks, the neural toolbox is the necessary equipment for every trading framework.

Package ats.fuzzy

We have presented in the sixth chapter basic principles of the fuzzy logic and fuzzy inference mechanisms. These principles are implemented in a fuzzy controller in the package `ats.fuzzy`. The construction and usage of fuzzy controllers in trading strategies has been conceptually sketched in Section 6.2. Now, we will take a look from the implementation point of view. Code A.4 shows how to construct a new fuzzy controller according to given rules and membership functions.

Code A.4:

```
FuzzyController c = new FuzzyController(DefuzzyMethod.COG, false);
// input variables
Variable v1 = new Variable(c, Variable.Type.INPUT, "VAR1", new Range(-1,1));
v1.addMf(new Trimf("MF1", -1.5, 0, 0.5));
...
```

```
// output variables
Variable v3 = new Variable(c, Variable.Type.OUTPUT, "VAR3", new Range(-1,1));
v3.addMf(new Trimf("MF3", -1.5, 0, 0.5));
...
// inference rules
FuzzyRule rule = new FuzzyRule(c, FuzzyRule.Operator.AND);
rule.addIf("VAR1","MF1", FuzzyRule.Negator.NORMAL);
rule.addIf("VAR2","MF2", FuzzyRule.Negator.NORMAL);
rule.addThen("VAR3","MF3");
...
c.validate();
```

The fuzzy controller is then used in a trading strategy as shown in Code A.5:

Code A.5:

```
Hashtable<String, Double> inputCrisps = new Hashtable<String,Double>();
inputCrisps.put("VAR1", crispValue1);
inputCrisps.put("VAR2", crispValue2);
...
Hashtable<String, Double> outputCrisps =c.compute(inputCrisps);
Double crispValue3 = outputCrisps.get("VAR3");
```

Package ats.simulation

Package `ats.simulation` implements a trading simulator based on the DEVS formalism. Additionally, it contains also various time and price slippage models as well as the interpolation models for different types of orders. Since the common principles have been introduced in the fourth chapter of this thesis, we won't go into particulars. We also don't think that it is necessary to explain the implementation of DEVS components and interconnections.

Because of this, we show only how to run simulations of trading strategies on given market data. Let's suppose that we have an array of market feeds[36], trading strategy, configuration for the broker/order execution component and the trading account. The simulation process is then invoked as shown in Code A.6.

Code A.6:

```
public HashSet<TradeReport> simulation(ArrayList<Feed> feeds,
                                       Strategy strategy,
                                       ConfigBroker broker,
                                       Account account) throws Exception
{
   Environment e = new Environment();
   e.setCommonAccount(account);

   for (Feed feed : feeds)
      e.addTradingThread(feed, strategy, broker);

   return e.run();
}
```

Considering the Code A.6, we firstly create a new trading environment which wraps all components of the simulation model and all interconnections between them. Then, we assign

[36] Each data feed is based on a certain timeframe and corresponds to a certain symbol.

the common trading account to this model. The parallel run of multiple trades is implemented as shown in Figure 4.11 of this thesis. Because of this, every market feed corresponds to an isolated tetrad of interconnected components. In the implementation, these tetrads are constructed by the `addTradingThread` method of the trading environment (see Code A.6).

Finally, the simulation is run by the invocation of the `run` method. This method returns a set of trade reports, one for each of market feeds. These reports can be later aggregated or statistically analyzed.

Package ats.optimization

Optimization tools are implemented in the `ats.optimization` package. This package contains an algorithm for exhaustive examination of parameter space and the implementation of the genetic algorithm introduced in the fifth chapter of this thesis.
Since all optimization algorithms fetch the structure of parameters automatically from the trading strategy, their usage is very simple. The only thing we have to do is to define the fitness function which will be used in the optimization process (see Code A.7).

Code A.7:

```
ExhaustiveSearch search = new ExhaustiveSearch();
search.setInputs(feeds,
                 strategy,
                 broker,
                 account,
                 new Fitness() {
                    @Override
                    public double compute(Stats stats) {
                       return stats.equityHigh - stats.equityDrawdown;
                    }
                 }
                 );
ParametricStrategy optimizedStrategy = search.start();
```

Similarly, the optimization by a genetic algorithm looks as follows (Code A.8):

Code A.8:

```
GeneticAlgorithm ga = new GeneticAlgorithm();
ga.setInputs(feeds,
             strategy,
             broker,
             account);

ga.configure(populationSize,
             combinationPercentage,
             mutationPercentage,
             mutationGenePercentage,
             numberOfIterations,
             new Fitness() {
                @Override
                public double compute(Stats stats) {
                   return stats.equityHigh - stats.equityDrawdown;
                }
             }
ParametricStrategy optimizedStrategy = ga.start();
```

A.3 Implementation of case studies

This thesis contains several independent case studies, which are discussed in individual chapters. Since all these studies have been implemented also in the trading framework, this section contains a complete list of them with references to the code.

Section:	1.2
Case study:	Attractors of daily and weekly price changes
Implementation:	`ats.caseStudy.csatractors.CSAttractor`

Section:	4.7.1
Case study:	Simulation case study
Implementation:	`ats.caseStudy.cssimulation.CSSimulation`

Section:	5.1.2
Case study:	Exhaustive optimization of the parametric trading strategy
Implementation:	`ats.caseStudy.csoptimization.CSExhaustiveOptimization`

Section:	5.3.3
Case study:	Optimization of the parametric trading strategy by a genetic algorithm
Implementation:	`ats.caseStudy.csoptimization.CSGeneticOptimization`

Section:	6.2.2
Case study:	Fuzzy surfaces of the fuzzy-based Triple Screen Trading System
Implementation:	`ats.caseStudy.csfuzzy.CSDisplayFuzzySurfaces`

Section:	6.3.1
Case study:	Sensitivity analysis of the whole market system (option 1)
Implementation:	`ats.caseStudy.csfuzzy.marketSystem.param1.CSFuzzySearch`

Section:	6.3.1
Case study:	Sensitivity analysis of the whole market system (option 2)
Implementation:	`ats.caseStudy.csfuzzy.marketSystem.param2.CSFuzzySearch`

Section:	6.3.2
Case study:	Sensitivity analysis of the conventional signaling system
Implementation:	`ats.caseStudy.csfuzzy.signalingSystem.CSConventional`

Section:	6.3.2
Case study:	Sensitivity analysis of the fuzzy-based signaling system
Implementation:	`ats.caseStudy.csfuzzy.signalingSystem.CSFuzzy`

Section:	6.4
Case study:	Simulation of the conventional Triple Screen Trading System
Implementation:	`ats.caseStudy.csfuzzy.CSConventionalTripleScreen`

Section:	6.4
Case study:	Simulation of the fuzzy-based Triple Screen Trading System
Implementation:	`ats.caseStudy.csfuzzy.CSFuzzyTripleScreen`

Case study: K-means based detection of Elliott waves
Implementation: `ats.caseStudy.cselliott.CSElliottKMeans`

Case study: Detection of Elliott waves based on genetic algorithms
Implementation: `ats.caseStudy.cselliott.CSElliottGenetic`

A.4 Code metrics

The following table contains code metrics of the framework implementation. Please note that all these values are only approximations.

	Total number of			
	packages	public classes	lines of code	kB of code
ats	1	1	159	5.9 kB
ats.caseStudy	11	36	3694	144.0 kB
ats.chartingElements	1	9	1158	43.8 kB
ats.database	2	6	518	22.5 kB
ats.elliottGenetic	6	12	1371	52.7 kB
ats.elliottWaves	4	14	2618	103.8 kB
ats.exception	1	9	288	10.3 kB
ats.formatters	1	9	353	13.8 kB
ats.fuzzy	1	7	437	15.4 kB
ats.gui	11	37	5926	248.2 kB
ats.neuralNetwork	1	4	691	32.3 kB
ats.optimization	4	8	770	27.4 kB
ats.out	1	1	86	3.3 kB
ats.simulation	16	51	4070	166.4 kB
ats.statements	1	4	564	28.0 kB
ats.strategy	6	18	1172	44.3 kB
ats.sync	9	18	2340	109.5 kB
Total	77	244	26215	1071.6 kB